Personalization Project 1 Report

Team members: Addison Li, Chuqi Yang, Haozheng Ni, Haoran Li

1. Introduction:
    1) Industrial background;
    2) Business objective;
    3) Dataset summary;
2. Technical approach;
    1) Preparation;
    2) Modeling details:
        i. Latent Factor model;
        ii. Hybrid model: User-based KNN Collaborative Filtering and past user transaction behavior;
    3) Prediction results and evaluation;
3. Real world application;
4. Future work and room for improvements.

Part I: Intro:

Industrial background:

As recommendation systems are widely applied in E-Commerce, we selected the dataset from RetailRocket, a SAAS company which designs recommendation systems to help e-commerce companies to provide better recommendations for the end consumers. RetailRocket's customers include Vans, Nespresso, and Groupon, to name a few.

Business Objective:

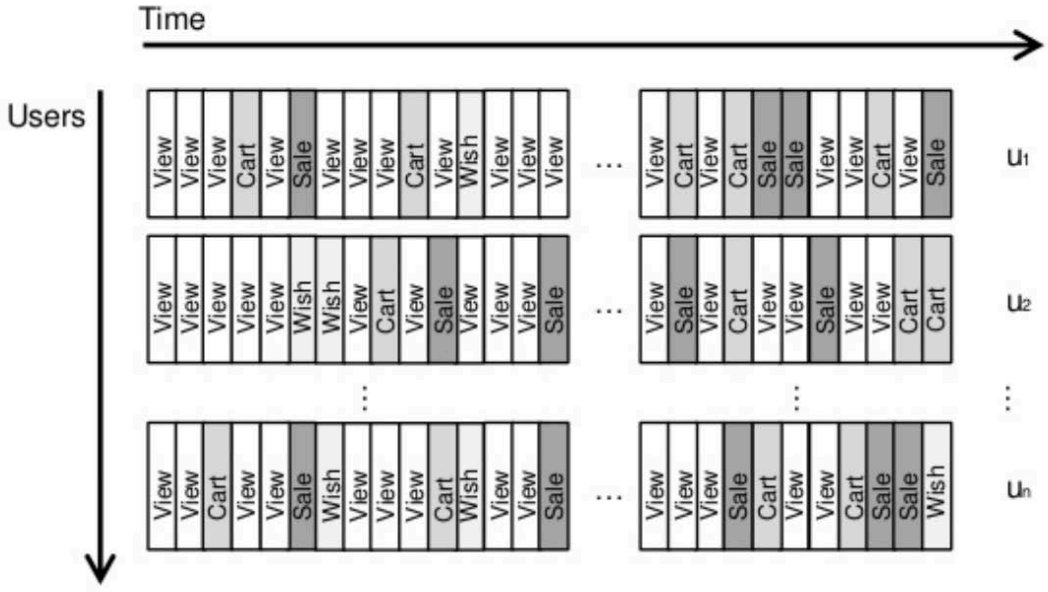
Positioning ourselves as the product managers and data scientists at RetailRockets, our goal is to help our customers, ie: E-Commerce companies to increase their sales revenue. We designed our product as a recommendation trio package, which is consisted of email, web, and mobile recommendations, with a core of double-model recommendation system.

In order to reach this goal, we separated the end consumers and strategized differently.

1, For new customers who we have little preexisting data of, we recommended them the most popular items from the data we collected from the existing shoppers.

2, For existing customers, we aim to increase their purchase by improving recommendation relevancy, enhancing shopping experience, and expanding customer retention with our recommendations. Our two models are: 1) Latent Factor Model; 2) Hybrid model: User-based KNN Collaborative Filtering and past user transaction behaviors. They work well and complement one another to make appropriate recommendations for these consumers.

Dataset overview:



The raw data includes 1,407,580 users, and 235,061 items. It contains: userID, itemID, categoryID, transactionType, ParentID, and Time Stamp. This data set does not come with explicit rating records. Therefore, we make predictions based on implied preferences reflected by users' behaviors.

We first eliminated Time Stamp and ParentID to move forward with our exploratory analysis. We didn't consider this factor because the quantity of the purchase, i.e., the time stamps per user is very sparse, simply not sufficient to provide reasonable conclusions.

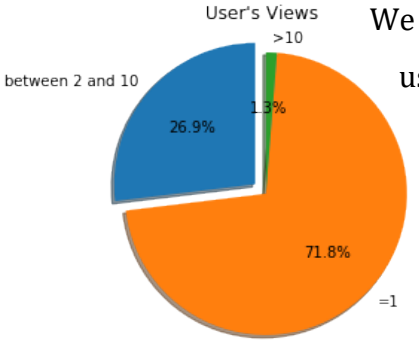
ParentIDs became insignificant after our items were aggregated under the main categoryIDs. Therefore, neglecting them won't influence our prediction. Due to the differences in implementation, more details on cleaning, sub-setting, and data manipulation will be explained in depth in the following technical section.

Part II: Technical Approach:

a) Preparation;
b) Modeling details:
   i. Latent Factor model;
   ii. Hybrid model: User-based KNN Collaborative Filtering and past user transaction behavior;
c) Prediction results and evaluation

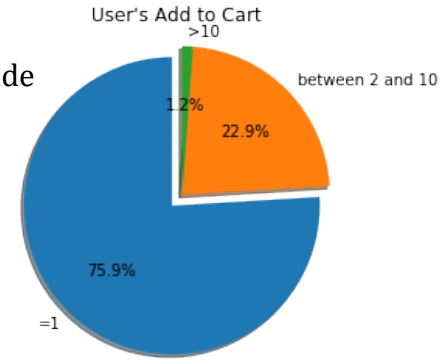
Preparation:

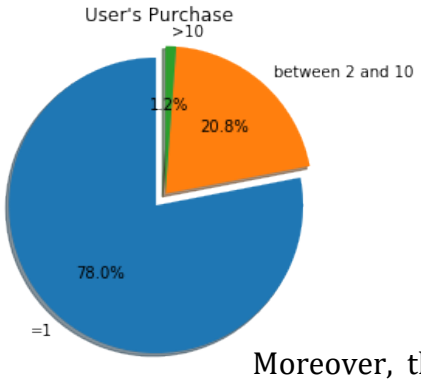
- Data cleaning and segregation – for new customers:



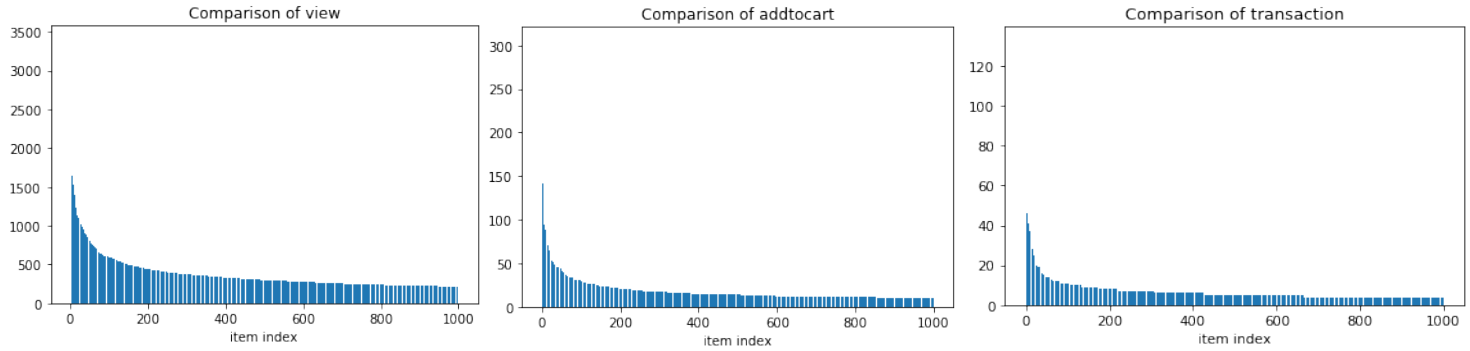
We first distinguished our users into two groups, new users, and existing users. Since we don't have any information about the new users' personal preference or shopping behaviors, we decide to recommend them the most popular items based on our existing customers' shopping history. Since there is no subsequent data about how these new users behave after our recommendations, we have yet any method to test the conversion rate of this group. Therefore, we focused on our existing users.

Most users in our dataset are not active, most of them only made one views, add to cart or purchase activity – graph to the right. This observation leads to our sampling method of users in next section.



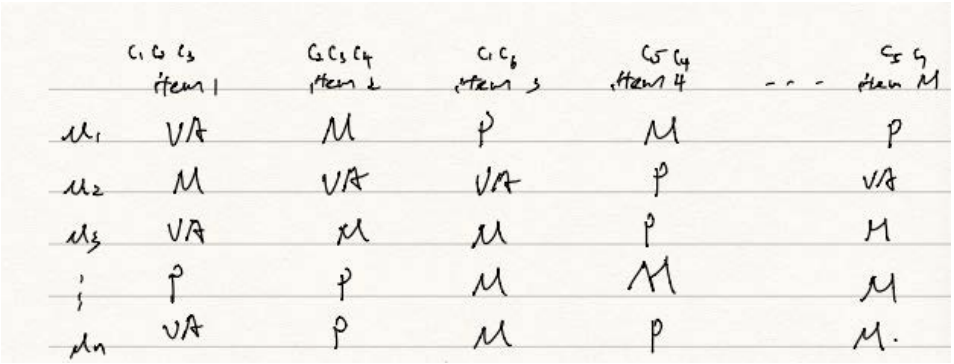


Moreover, the items have long-tail effect, meaning that some items are more popular than the rest, which motivates us to sample the non-popular items as well.



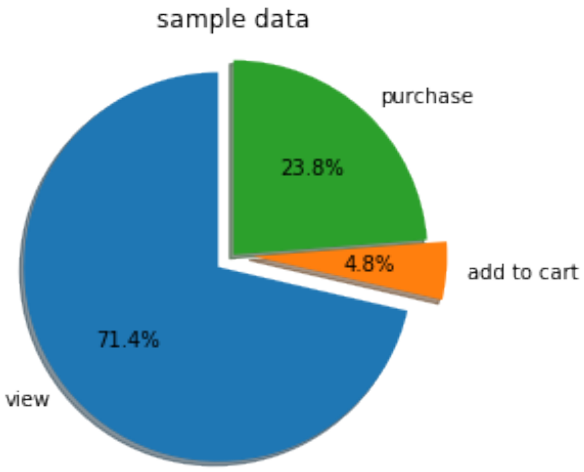
- Systematical sampling – for the existing users:

Due to the specific characteristics of our data, randomly selecting users or items won't provide enough quality data for us to process. To ensure enough data to conduct training and testing, we filtered our end consumers and only kept those who have at least three purchases and at least three views or add-to-cart activities, which yields to 1016 users and 235,061 unique items.



With 235,061 items and a significant amount of missing data, our user-item matrix is very

sparse. This is far from enough to make reasonable prediction. To address this issue, we grouped the items that belong to the same category after recording each user's behavior toward each item, transforming our user-item matrix into user-category matrix. Thus, our matrix ended up with 1016 users and 1962 categories. The proportions of view, add to cart and purchase are more balanced.



Through this transformation, the user will no longer get a recommended item from the exact subcategory that he or she had purchased from, but an item from the general category that he or she has shown activities in. As long as the users purchase anything we recommended, the overall sales revenue will increase and our business objective won't be compromised.

Modeling details:

As mentioned earlier, we built two models to serve our existing customers: Latent Factor Model and a hybrid model that combined User-based KNN Collaborative Filtering and past user transaction behavior.

- **Latent Factor Model:**

  Latent factor model uses latent vectors that represent inter-item relationships in lower dimension, which is ideal for our data set since the items are naturally grouped under sub-categories, indicating that we can make use of their relevancy for prediction

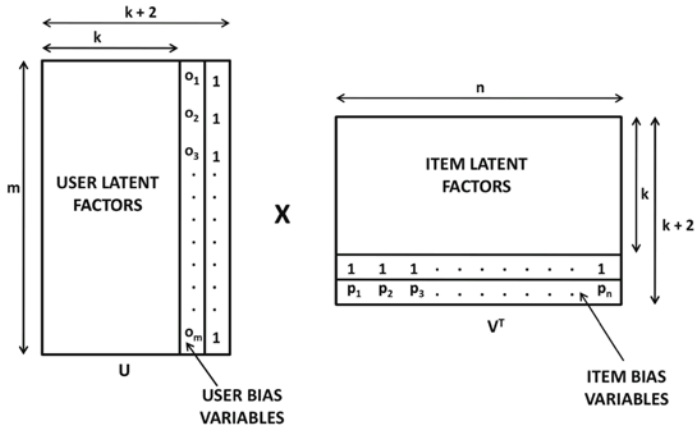
purpose.

Our experiment is conducted as follows:

Firstly, we incorporated user bias and item bias in our model. These two biases are two vectors of the existing users' shopping behaviors and biases toward different items. User bias represents the popularity of items based on information from the active existing shoppers. Item bias represents individual users' personal preference toward certain item.

The baseline model only contains those two biases. The mode is represented by the following formula and picture:

$$\widehat{r_{ij}} = o_i + p_j + \sum_{s=1}^{k} u_{is} v_{js}$$



Our target is to minimize the following by stochastic gradient descent method implemented by ourselves.

$$J = \sum_{(u_i, v_j, r_{ij}) \in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} v_{sj} \right) + \frac{\beta}{2} \left(\sum_{s=1}^{k} (||U||^2 + ||V||^2) \right)$$

Then, we divided our data into three parts for training (TR), cross validation (CV), and testing (TE) respectively, 80% for TR and CV, 20% for TE. We then assigned different weights to our categorical data M, V, A, P according to their hierarchical influence over purchasing decisions, transforming them into quantitative values such as 0M, 1V, 2A,

5P. In the training data, Stochastic Gradient Descent is used to tune different sets of parameters through rounds of cross validations.

To find out the best parameter set, we used Hit Rate and Precision as the main metric for this model. The Hit Rate is calculated as follows:

$$Hit\ Rate = \frac{the\ \#\ of\ recommended\ items\ \cap\ the\ \#\ of\ Purchases\ in\ TE}{the\ \#\ of\ recommended\ items\ \cap\ total\ records\ in\ TE}$$

For every user, we make a total of 50 recommendations. We then count how many of these items appeared in the Test set. The number of items serves appeared in both sets serves as our denominator, while the number of items appeared in both recommendations and ended being purchased serve as our numerator.

Hit Rate is indispensible because it enables us to measure how close we can "understand" a user's preference, which increases the relevancy of our recommendations.

The other metric we look at is Precision, which represents the accuracy of our prediction. It is calculated as follows:

$$Precision = \frac{the\ \#\ of\ recommended\ items\ \cap\ the\ \#\ of\ Purchases\ in\ TE}{the\ \#\ of\ recommended\ items}$$

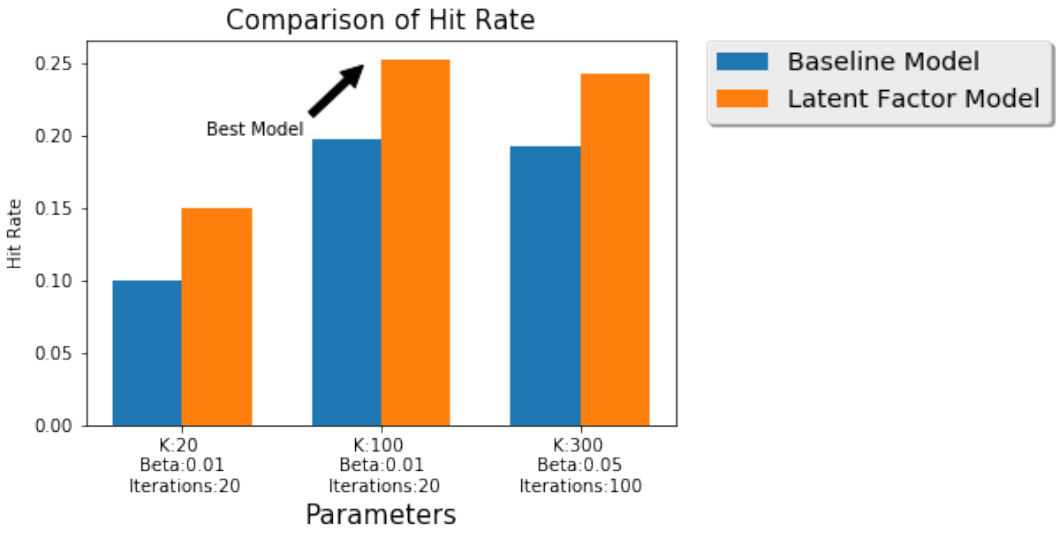
This is important because it indicates the percentage of how many items we recommended to the user ended being purchased, which could be directly translated into sales growth for our customer companies.

Finally, we picked the set of parameters, which generated the most accurate prediction results to examine the final Test set.

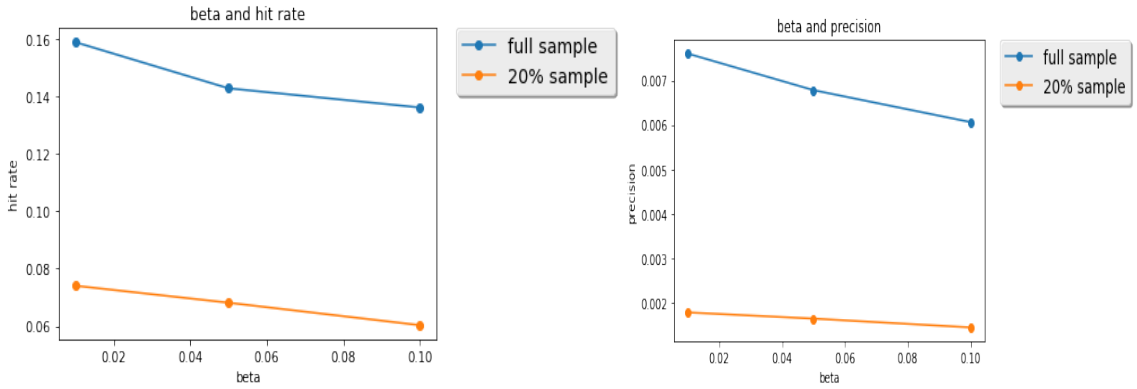
Result:

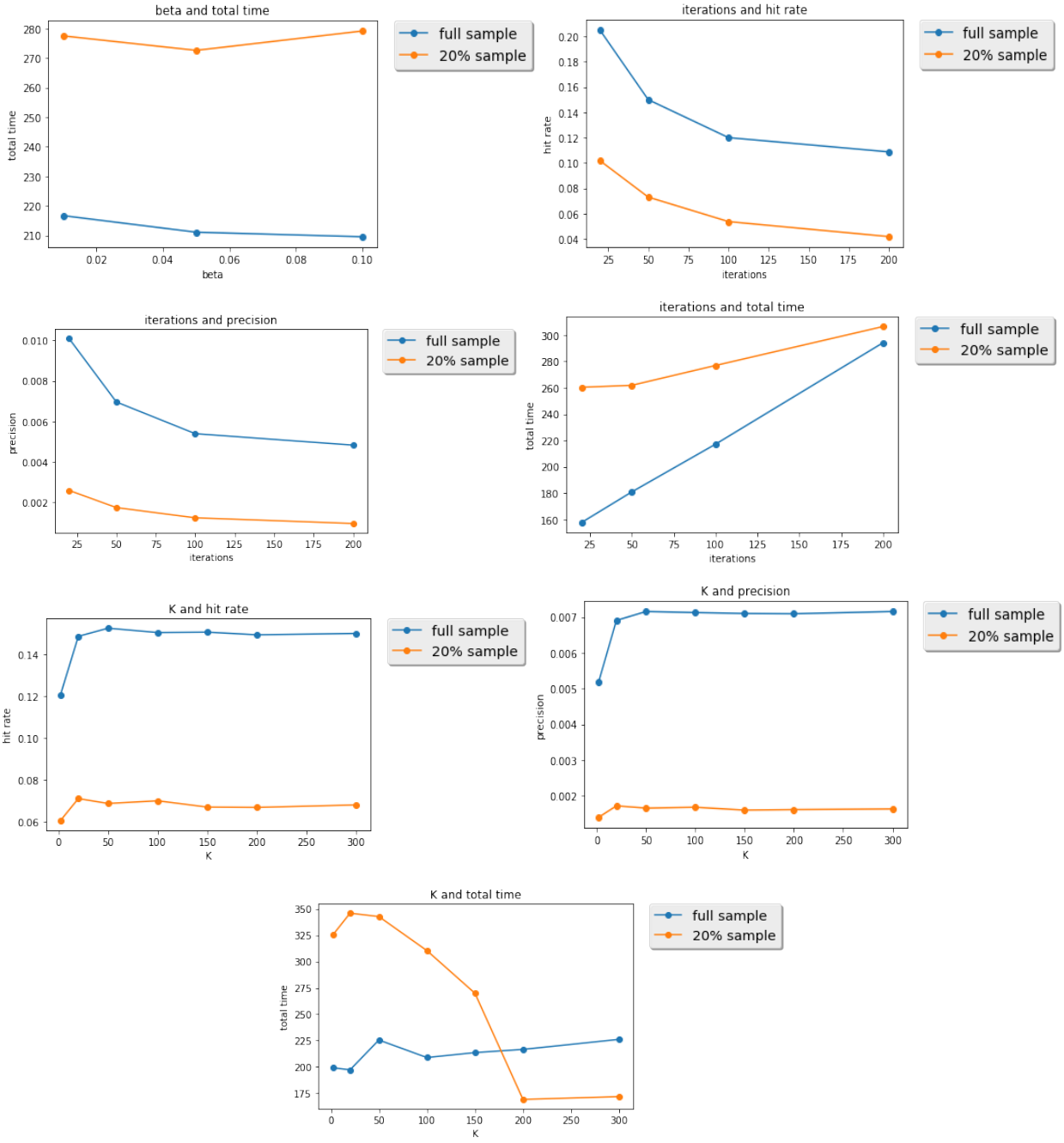
The best result we obtained from cross validation is 100 latent factors with regularization 0.01 and 20 iterations of Stochastic Gradient Descent. Compared with the baseline model and other parameter sets, this set yielded the prediction results of

the highest accuracy.



The following plots recorded how Hit Rate and Precision changed as each parameter changed, while other parameters remained fixed. The learning rate of Stochastic Gradient Descent is set to be 0.01 because any larger value can cause our data to overflow. To run through our entire data set, cross validation takes close to 10 hours. This time reduced drastically down for a smaller sample size with 20% data. In the graph above, K stands for number of latent factors, Beta stands for regularization term, and Iteration represents the iteration of Stochastic Gradient Descent optimization. In the graphs below, the run time comparison results for full and 20% sample size are displayed together.



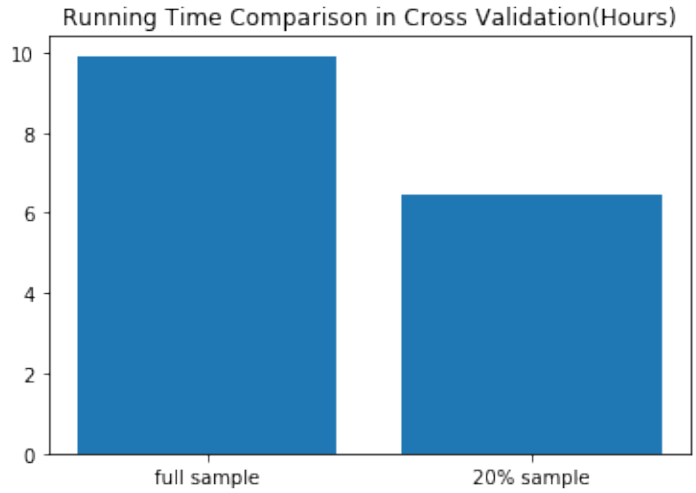


The changes in overall accuracy and evaluation metrics：

Based on the results, both full sample and small sample support that the prediction precision increases with the number of latent factor variables going up to some point, and a moderate K shall be selected to avoid overfitting. Havier regularization may not be helpful in increasing precision, while large number of iterations of Stochastic Gradient Descent will lead to overfitting.

Run-time VS data size:

As we can see from the graph blow, a sample of 20% of full size costs about 6 hours to run, while the full size data only costs a little under 10 hours. This means that our model can scale relatively well when data size increased by 5 times.



Other considerable designs:

Observing the plots of hit rate of different parameter, we could further cross validate our parameters with narrower range. For example, we could try the value of latent variables at 100, iteration at 20 and regularization term at 0.01, to see what kind of difference it would make.

Moreover, we can reduce the number of recommended items and see how that would affect our precision.

Lastly, we can try to initialize different values to each behavior to see how implicit feedback may change. SVD++ could be used too to further enhance the impact of implicit feedback.

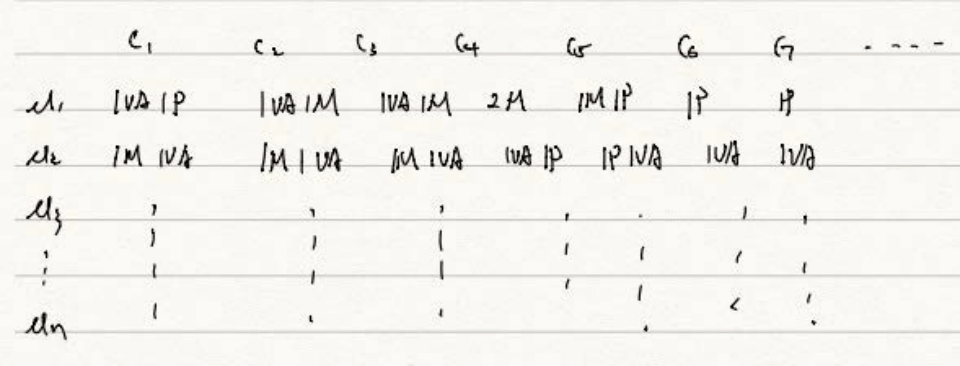
- **Hybrid Model: User-based KNN Collaborative Filtering + Past User Transaction behavior**

  KNN Collaborative Filtering is a classic application for user-based item recommendation by leveraging top-K most similar users. However, limited by lacking the explicit rating data, we strengthened this method by adding users' transaction behavior history. Thus, for each user, we not only look for the top-K most similar users,

but also took into consideration of heir purchase and browsing history to make more tailored predictions to each user.

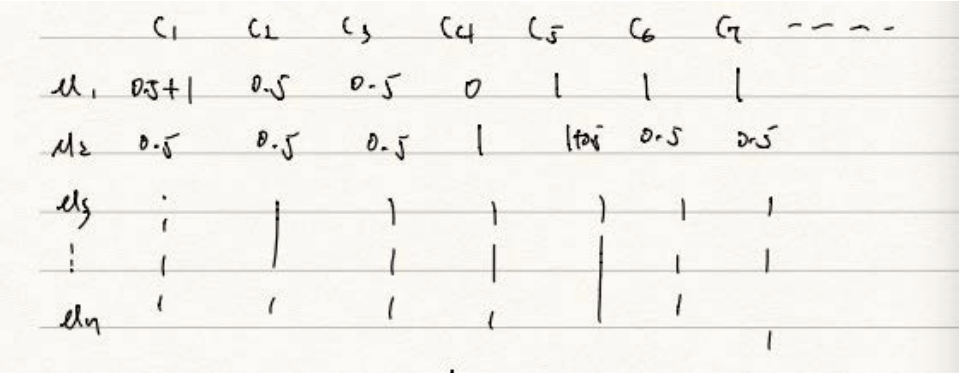
The process goes as follows:

We first constructed user-category matrix by transforming item preference to category preference. We combined "view" and "add to cart" as one behavior VA because for this model, they both indicate that the user would have a bigger purchasing potential. Since we don't have extra data to further distinguish their influence, we combined these two factors. For every user, we first grouped all the data points we have about the items in the same categories, and summed up the number of each of their behaviors, such as U_1C_1 = (20VA, 7P, 2M).



Assign weights:

Since P (Purchasing) is a stronger buying indicator than VA (View and Add to cart), we assigned the weight of 1 point for P, 0.5 for VA, and 0 for M (Missing). We then sum up all preference values for this user to arrive at a preference total per category:



Finally, we acquired percentages of interest:

We used the preference value of a single category divided by the total preference to get a percentage, which stands for this user's interest about the items in this category among all the potential categories of this user.

The next step is to subset our data for training, cross validation, and testing purposes.

1) Preliminary sub-setting into 80% - 20%:

   After the user-category matrix is formed, we transformed this matrix into a dictionary containing four columns, with each column dedicated to: "user – category – preference percentage – random variable ". We introduced a uniformly distributed random variable over the interval between [0, 1] to randomly subset our data into 20% and 80% respectively for testing and training purpose.

2) Reinsurance:

   By randomizing the users, we might end up having a training set and testing set containing different users, which would cause trouble for us to compare predictions. To rule out this possibility, we added secondary filtering criteria to ensure that the user data we used for testing must have appeared in the pre-existing training data. In this way, we are guaranteed to have sufficient data to train, cross validate, and test our results. Our final training set contains 1,603,819 records, and 389,573 records for testing set.

Calculation:

We started our similarity calculation with simple Pearson Correlation Coefficient:

$$Pearson(u,v) = \frac{\sum_{k \in I_u \cap I_v}(r_{uk} - E(u)) * (r_{vk} - E(v))}{(\sum_{k \in I_u \cap I_v}(r_{uk} - E(u))^2)^{1/2} * (\sum_{k \in I_u \cap I_v}(r_{vk} - E(v))^2)^{1/2}}$$

However, the PCC result ended up being 1 as a result that part of our users only have one category in common with other users. This is misleading because "1" can also indicate that two users has exact same shopping preference. To fix this undesirable coincident, we introduced weighted PCC and weighted Jaccard.

$$Sim(u,v) = 0.5 * \frac{\sum_{k \in I_u \cap I_v} \left(r_{uk} - E(u)\right) * \left(r_{vk} - E(v)\right)}{\left(\sum_{k \in I_u \cap I_v} \left(r_{uk} - E(u)\right)^2\right)^{\frac{1}{2}} * \left(\sum_{k \in I_u \cap I_v} \left(r_{vk} - E(v)\right)^2\right)^{\frac{1}{2}}}$$

$$+0.5 * \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

After calculating all the similarities between users, we started to predict our recommendations using:

$$\widehat{r_{iJ}} = E(u) + \frac{\sum_{u \in P_u(j)} Sim(u,v) * \left(r_{vj} - E(v)\right)}{\sum_{v \in P_u(j)} |Sim(u,v)|}$$

The result was disappointing because the formula above incorporated the average of users' ratings, which is not applicable for our model since we used an interest percentage to reflect the users' preference, rather than a rating system. By getting rid of the rating average U, we were able to acquire a more reasonable calculation.

$$\widehat{r_{iJ}} = \frac{\sum_{u \in P_u(j)} Sim(u,v) * \left(r_{vj}\right)}{\sum_{v \in P_u(j)} |Sim(u,v)|}$$

Last but not the least, we incorporate two metrics to measure this model: MSE, and hit rate.

$$MSE\ for\ an\ individual\ user\ u$$

$$= \frac{1}{|rated\ categories|} \sum_{rated\ categories} (\widehat{r_{ui}} - r_{ui})^2$$

$$Total\ MSE = \frac{\sum_{All\ users} MSE\ for\ an\ individual\ user\ u}{n}$$

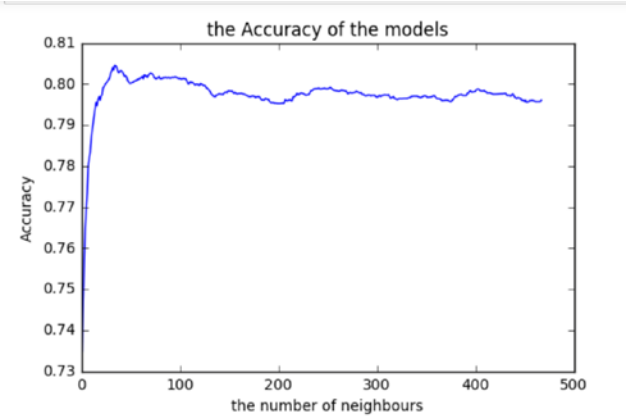
$$Accuracy\ Rate\ for\ an\ individual\ user\ u =$$

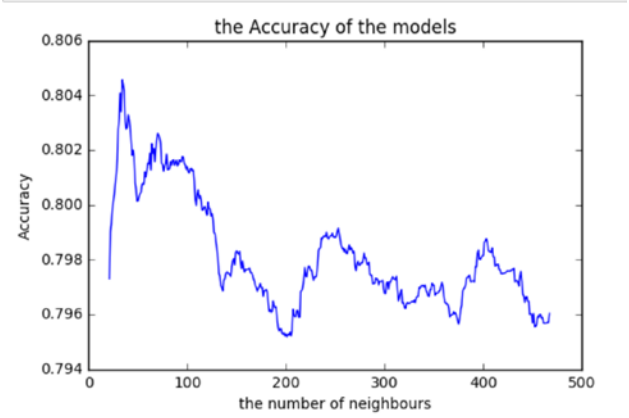
$$\frac{Number\ of\ the\ overlapping\ top\ \min(20, number\ of\ interested\ categories)\ recommended\ categories\ and\ top\ \min(20, number\ of\ interested\ categories)\ most\ interested\ categories}{\min(20, number\ of\ interested\ categories)}$$

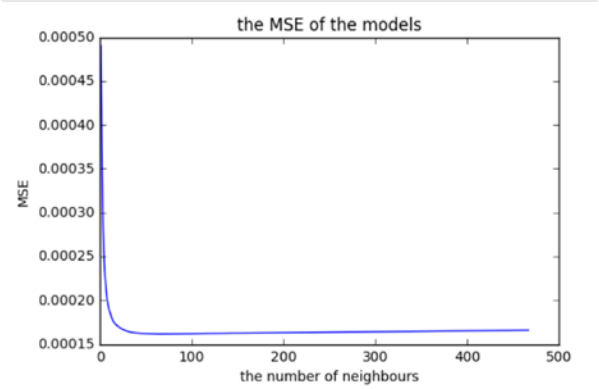
$$Total\ Accuracy\ Rate = \frac{\sum_{All\ users} Accuracy\ Rate\ for\ an\ individual\ user\ u}{n}$$

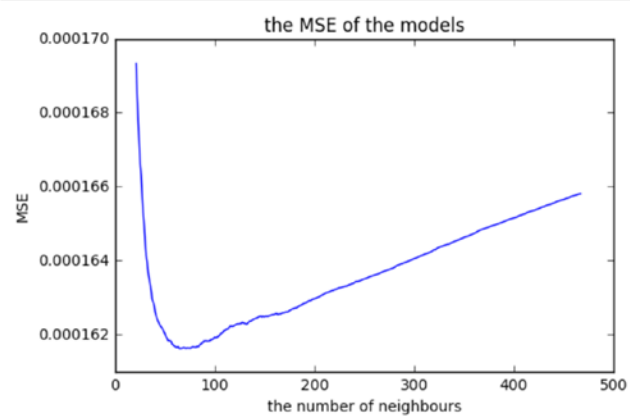
We eventually tuned parameters and found two sets of K values, 65 nearest neighbor would generate the smallest MSE and 34 nearest neighbor would maximize hit rate.

Following plots shows the results in cross validation, the left columns are plots for all values of k and the right column remove first several values because they perform really bad.



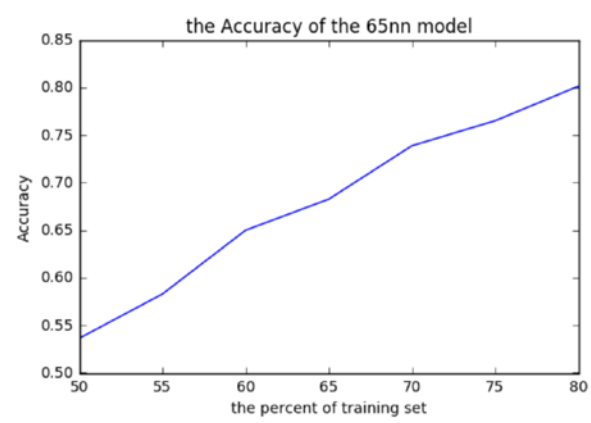


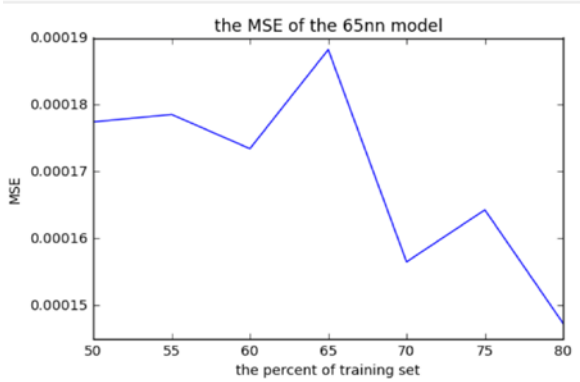




Scalability:

Based on the best two sets of parameters we have found, we tested the results by splitting training and testing data in different proportion, from 50% - 80%. The Accuracy below indicates the Hit Rate. The corresponding MSE and Hit Rate are plotted as follows:





In brief, the larger the number is of the training set, the better the Hit Rate, which is consistent with our intuition such that the more data we have about our neighbors, the more accurate the simiarirties would be. However, MSE's plots showed strange patterns as the number of neighbors in the training set increases to 34 and 65. It seems that moderate size of data does not strictly outperform smaller data, and the reason of this observation is not clear yet.

PART III : Real World Application

As Product Managers and Data Scientists at RetailRockets, our primary goal is to design recommendation systems that could help our customers – E-Commerce firms to increase sales revenue. Our models can be integrated with a wide range of applications. We'll use marketing email campaigns, web-recommendations, and mobile in-app recommendation as our main application focus.

Marketing email campaigns incorporated with tailored recommendation solution can be used for both reaching out to new and existing customers. For new customers, emails

sharing the most popular items and shopping trends could generate interest and help engage with these users. For the existing users, relevant recommendation emails can serve as a reminder for them to reorder regularly, or explore cross-sale opportunity.

Our model is also well suited for web recommendation. By improving the prediction and recommending accuracy, we help our customers' users discover about their needs, reduce the potential bouncing between competitor websites, and mitigate the risk of loosing customers.

Lastly, our models can also be integrated with mobile in-app recommendation. As mobile usage and accessibility is growing rapidly among increasing numbers of users, mobile app recommendation becomes indispensible. Integrating with a simplified check-out process and mobile-app notification, in-app recommendation can help companies increase customer loyalty and stickiness, enhance shopping experience, and improve customer satisfaction.

By bundling these three applications, we can create a holistic and convenient one-stop shopping experience for the end consumers. As a result, this will maximize the sales revenues for our customers, making our product more valuable.

From a cost saving perspective, our customers don't have to invest heavily in engineering talents, development time, and potential opportunity cost. The more customer success cases we can build to establish our reputation, attract more potential customers, and expand our market share.

By providing a valuable core product, we can save our marketing budget by winning customer referrals. An expansion in market share and customer quantity will also contribute to our company evaluation, attract more funding, and stimulate our growth.

PART IV: Future work

- Limitations:

While we are confident about our model and prediction results, we did encounter some serious limitation along the process.

Initially, we don't have any information about our items, or categories. It makes it hard for us to consider the intrinsic differences between categories. For example, cars sales and

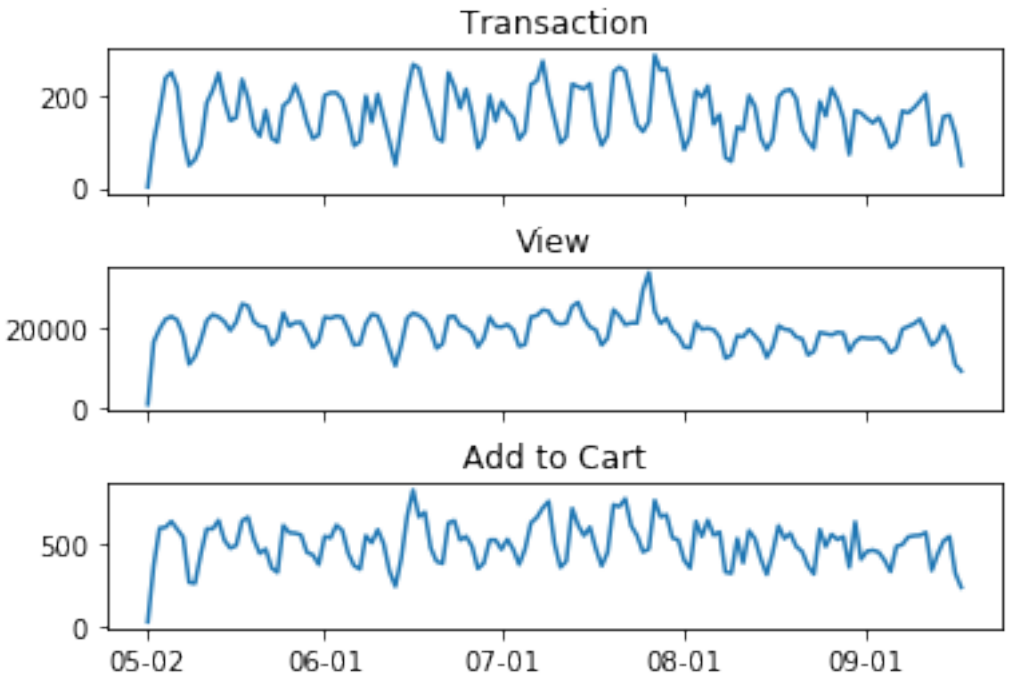
grocery sales can yield to totally different recommendation strategies. If this information is provided by RetaiRocket, we could have better tailored our models for specific industries.

In addition, all of our data points about users' feedback are implicit, rather than direct ratings. When transforming implicative information into real-values, we mostly based on our experience or intuition, which is not the systematic, nor scientific without mathematical support from real data.

Lastly, sparse data set remains one of the most common challenges. It requires most careful consideration when deciding on a strategy to fill up the missing data. Especially for E-Commerce, the more data we have for an individual user, the more we can understand this user's interest therefore predict his next shopping behavior.

- Future work:

We'd like to distinguish short-term and long-term recommendations leveraging Time Stamps.



One of our data categories we did not make use of is Time Stamp. The reason was that there were not enough time stamps for a single user to be further analyzed. If we have time stamps for a longer period of time, where we have multiple entries for a user, it would be

interesting to see how a user's shopping preference might vary over different time periods. From above plot of time series, we can observe some similar pattern of users' view, add to cart and purchase. From marketing standpoint, we could and should develop different campaigns targeting shoppers at different times.

Another interesting hypothesis is that consumers may have different shopping styles: some always stay with their usual choices, while others are willing to explore new things. Knowing this difference, we could develop different recommendation strategies to for the diverged groups.

Last but last the least, our recommendation system design could also line up with our customer companies' business priority. We could incorporate inventory and price information to maximize profit for our customer companies.

Summary

From the results of both of our models, we feel confident to put our package in a real company as a product. Overall, compared with baseline model, our product has increased prediction accuracy with an acceptable runtime and scalability. With more information about our data, categorical information, different industry backgrounds, further tuning with our parameters, we would be able to modify our model and enhance our performance further, and create more value for our customers.