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

Project Part II:

New feature: Time SVD for Latent Factor Model:

We chose Time SVD as the new feature to be added to our latent factor model. The motivation behind Time SVD is that users' preference is changing over time. We need to have a recommendation strategy that reacts to this change. In addition, other external factors, such as newly added items and shifting in trends can also influence the popularity of items. Therefore, by adding Time SVD, we incorporated these observations, making one step further than conventional SVD to model item factors and user factors as functions of time. As we were designing our feature, we referenced the paper by Y. Koren[1], which proposed the method of adding Time SVD++. However, since our data naturally contains implicit feedback, we removed the implicit matrix decomposition from the model. To reflect the implied relationship, we used 1,2,5 to represent and computed "view", "add-to-cart", and "purchase" numerically.

In SVD, the baseline model that models user and item effect is defined as:

$$b_{ui} = \mu + b_u + b_i$$

where b_u, b_i are user and item effect respectively. Now we regard them as a function of time, i.e:

$$b_{ui}(t) = \mu + b_u(t) + b_i(t)$$

The author made another assumption that user effect can fluctuate in very short period of time, and thus $b_u(t)$ is modeled on the daily basis. While the effects of items may not change frequently, we divided the time periods into different bins and $b_i(t)$ remained unchanged in each bin. So we further define

$$b_i(t) = b_i + b_{i,Bin(t)}$$
$$b_{u(t)} = b_u + \alpha_u * dev_u(t)$$

where $dev_u(t)$ measures how much current time stamp differs than the median time stamp, i.e

$$dev_u(t) = sign(t - t_u)|t - t_u|^\beta$$

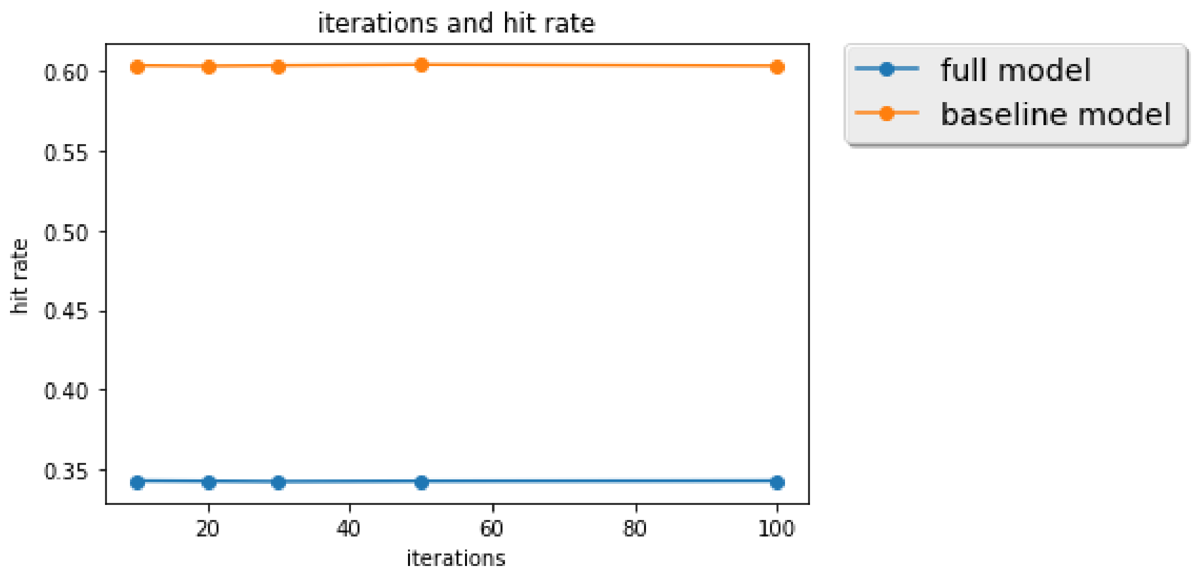
Now we are ready to define the full model, where user factors were added to the baseline model above.

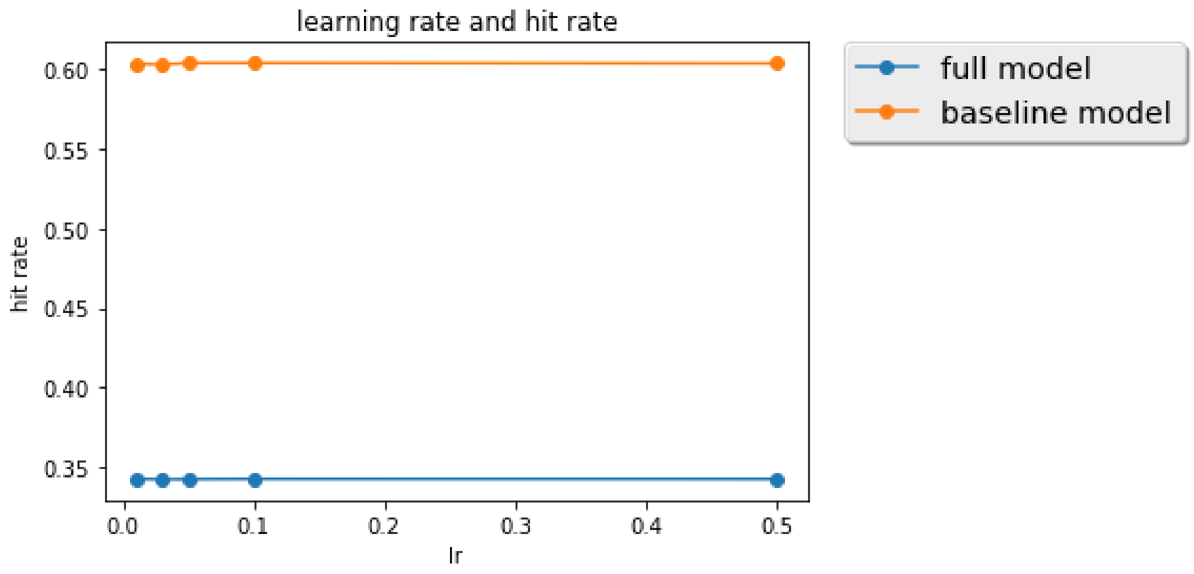
$$r_{ui}(t) = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$$

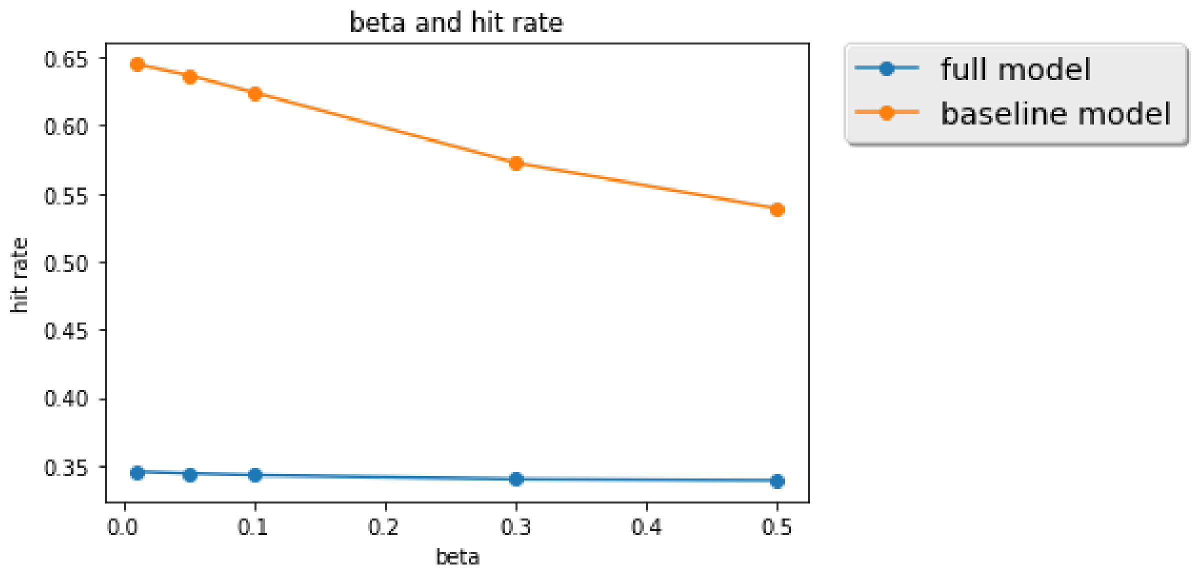
Since there are no available Time SVD++ library in Python, we wrote our own and used stochastic gradient descent method to estimate parameter. We tried cross validation on different numbers of latent factors, numbers of bins($nBins$), power in deviation(β), learning rate, and iterations in sgd. The plot against hit rate is shown below, and the hit rate is defined as proportions of items among 20 recommendations that were eventually purchased.

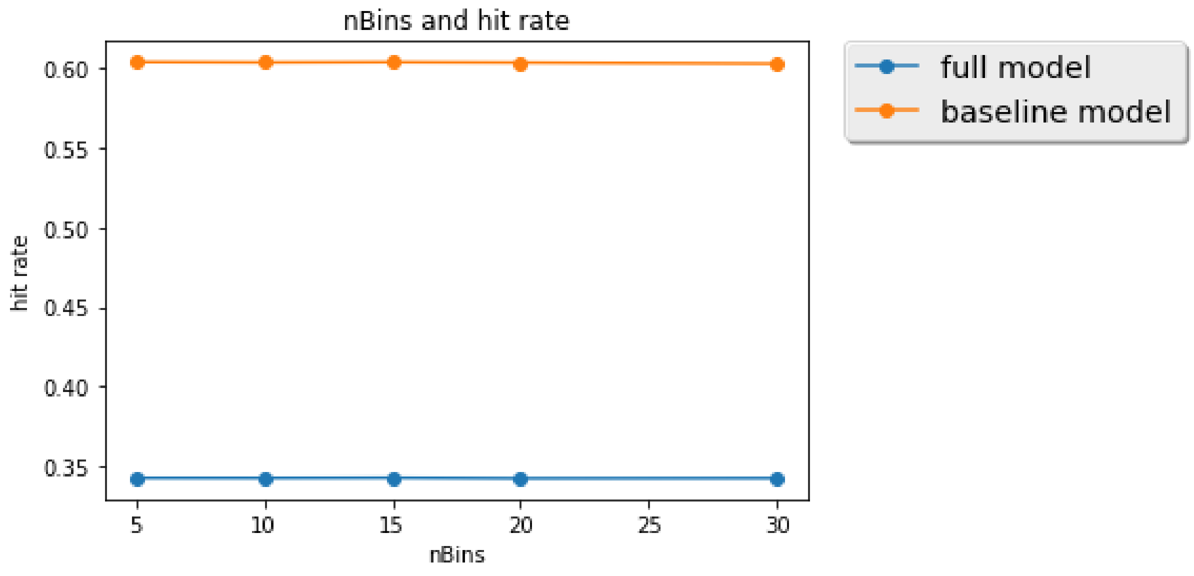
The baseline model only contains the user factor and item factors, without adding the interaction with them, i.e

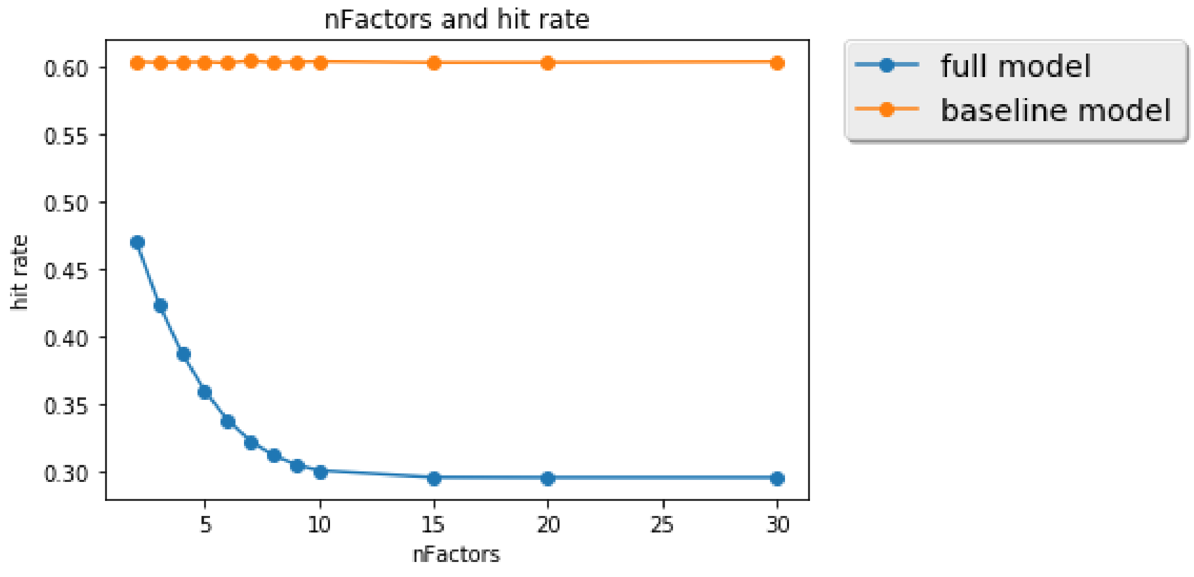
$$r_{ui}(t) = \mu + b_i(t) + b_u(t)$$



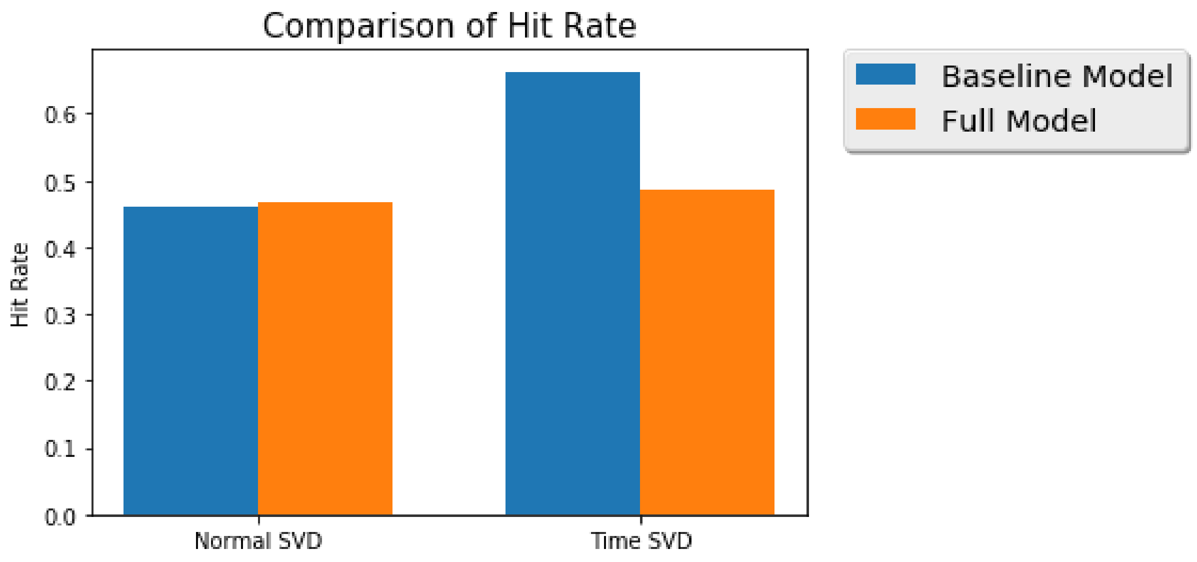



beta and hit rate
hit rate
beta
full model
baseline model
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.0
0.1
0.2
0.3
0.4
0.5

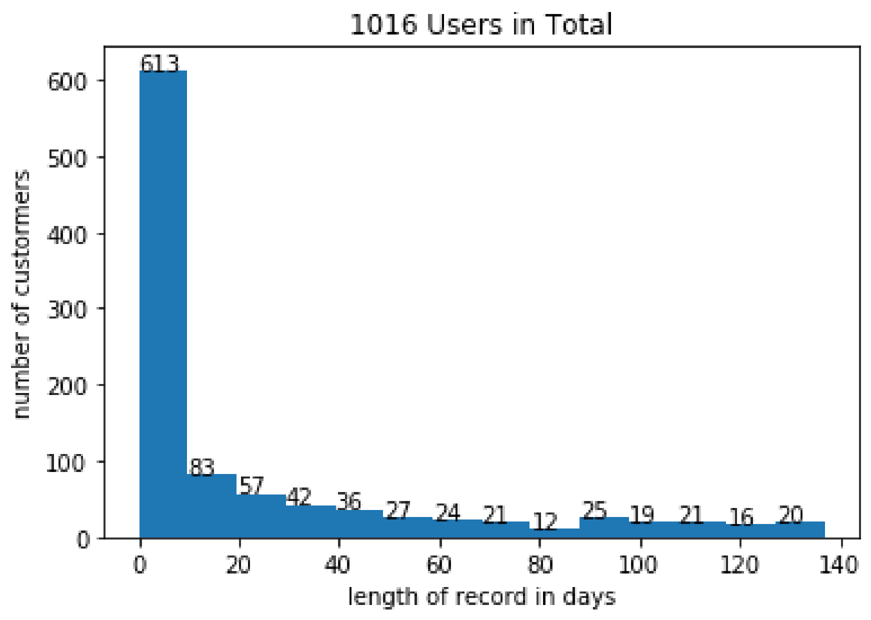

nBins and hit rate
hit rate
nBins
full model
baseline model
0.60
0.55
0.50
0.45
0.40
0.35
5
10
15
20
25
30


nFactors and hit rate
hit rate
nFactors
full model
baseline model
0.60
0.55
0.50
0.45
0.40
0.35
0.30
5
10
15
20
25
30

Notice that in splitting the training and testing sets, we should make sure that no records in the test set appear earlier than the train set, as we are trying to learn the behavior through time. It is clear that the general baseline model behaves better than the comprehensive model in the cross validation set. The number of latent factors is a critical parameter for the new comprehensive model. It is also interesting that the number of bins does not affect too much in both models, which is quite against the intuition as items shall yield changing popularities through time and different splits of time line should capture different patterns. We made more investigation later.
We compared the time SVD and conventional SVD predictions on the test set, with the best parameters obtained from validation set.



From the plot, it is obvious that the comprehensive model of Time SVD performs similarly as conventional SVD, and the baseline model is better than the rest. This supports the conclusion in the paper that time SVD does has its advantage when we have ample information on time stamps.
To better understand why the comprehensive model does not outperform the baseline model, we analyzed the length of records in days for each customer, the plot of which is shown below



Also, a big number of our customers only have activities within last ten days. This number is as high as 50% or even higher, which indicates that the time data on customers who have at least two purchases and two other activities are in fact very sparse. What's worse, up to 344 activities from these users were made in one day, which left very small room for modeling the behaviors through time. The paper used 200 bins to show best result.

The requirement on long period of time, and moderate size of data per customer to train the factor is one major limitation of time SVD. This observation explains why the comprehensive model does not outperform baseline model.

Many further improvements can be made if we have more data for each customer in a longer period of time. For example, we can amplify the effect by raising to exponential

$$b_{u(t)} = b_u + \frac{\sum_{l=1}^{k_u} e^{-\gamma|t-t_l^u|} b_{t_l}^u}{\sum_{l=1}^{k_u} e^{-\gamma|t-t_l^u|}}$$

Moreover we can incorporate the change of behaviors through time cycle, and maybe regularization the decomposition can also be helpful.

Improvements: User Behavior & Neighborhood Hybrid Model:

In the first part of our project, we designed a hybrid model that generates recommendation by weighing an individual user's past purchasing behaviors and behaviors of this user's top similar users. The weighted result determines which items to be recommended to this user eventually. To fix potential training and testing sets dependency and to achieve higher prediction accuracy, we modified several steps of this weighted model.

1. Remove potential dependency between training and testing sets:

In the part I, we first calculated a "percentage of interest", a percentage score which represents an individual user's interest about the items in a certain category among all the categories. We then divided the testing and training sets into subsets. However, this could lead to dependency between our training and testing sets and cause a fundamental issue. As the items were merged into categories and the "interest percentage" scores calculated before testing and training subsets are made, our testing set could become subjected to inaccurate influence and then produce results with undesirable bias. Therefore, to remove this bias, we divided our training and testing datasets first and then calculated the user's "percentage of interest".

2. Further tuning parameters in search for better performance:

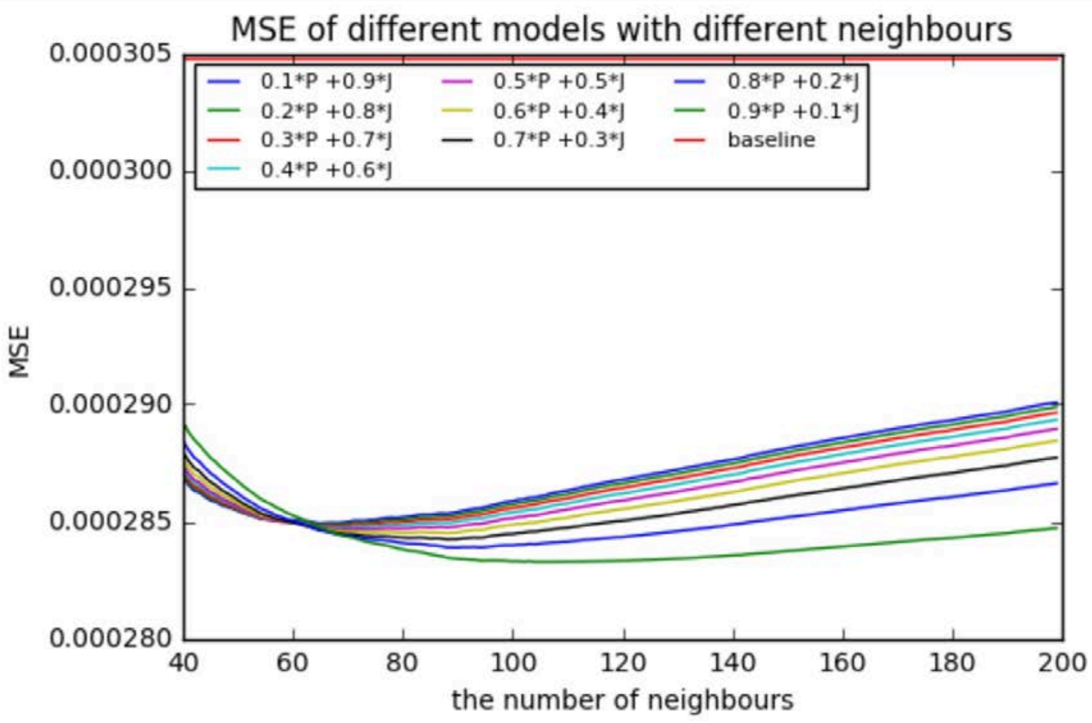
In part I, the weight parameters were initially set as 0.5/0.5 to reduce the calculations workload. 0.5/0.5 is also convenient as we believed that a user's past purchase behaviors and his/her similar users' behaviors play an equal weight in determining our recommendations. However, this may not be the case. In search for better performance, we trained 9 sets of parameters and eventually adjusted our weights to 0.9 for modified PPC and 0.1 for Jaccard, respectively, which produced the smallest MSE. After applying different PPC values and calculating all the scores for the training set, we combined them with the scores in testing set, which gave us a complete list of interest scores for a certain user regarding all the categories. We then picked the top 20 and recommended items from these categories.

3. In project II, we further segmented our entire user space to three categories and applied different recommending strategies:
   a) Type A: Users who have at least three purchase transactions and three views or add-to-cart activities, but has a purchase time scope within a year. For this group, we recommended items using the User Behavior & Neighborhood Hybrid Model described above.
   b) Type B: Users who have at least three purchase transactions and three views or add to cart activities, but has a purchase time scope over a year. For this group, we recommended items using the Time SVD model
   c) Type C: Users who have no purchasing activities but some browsing activities. For these users, we pushed the top popular items from the general popular item list.

d) Type D: The rest of the users. For this group, we pushed the top popular items that are from the same categories as the items that this user browsed or previously purchased.

Results:
(notice: in part I, our model did not outperform the baseline model)
The baseline model is defined set forth below:
From the training set, for each category, we took the average of the computed user interest score. We assigned these average scores to the corresponding categories for all users in the testing set. For each user, we recommended the highest 20 user interest score corresponding categories. Essentially, we recommended the top 20 most popular items across all users.

- MSE:



We define our MSE as the following:

$$MSE\ for\ an\ individual\ user\ u = \frac{1}{|rated\ categories|} \sum_{rated\ \ categories} (\widehat{r_{ui}} - r_{ui})^2$$

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

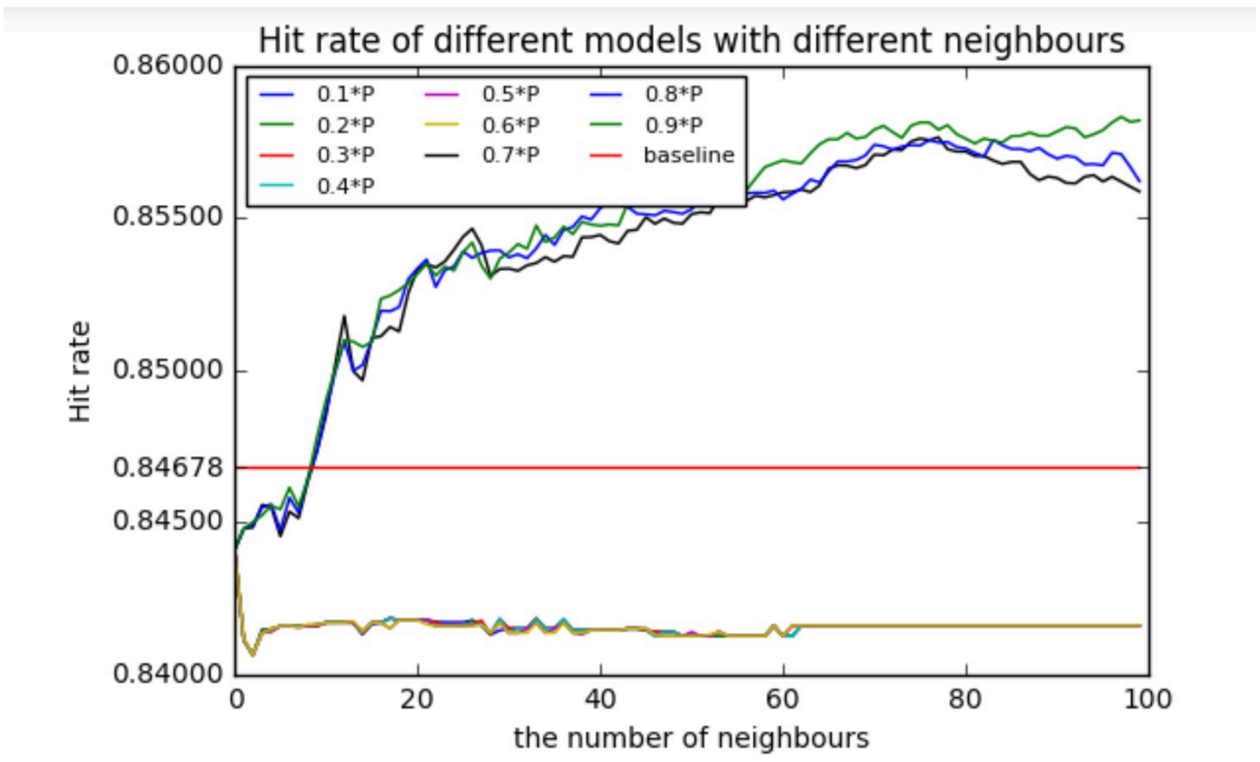
(note: For the baseline, if for a category in the testing set, there is no interest score in the training set, we set the testing set score equals to 0)

This MSE graph captured how accurate we were able to predict different users' interest for certain category and give a score. From the above graph, regardless of the weight, our model is clearly better than the baseline MSE, which equals to 0.000305. Specifically, we can see that when we use 0.9*Modified Pearson+ 0.1* Jaccard, with 109 nearest neighbors, we are able to get the smallest MSE = 0.000283.

Our model result is better because it works this way: Since our data is very sparse, in fact that many of our users do not have many interested categories, therefore their interest rates for all other categories should equal to 0. However, the baseline model pushes the most popular items to the users based on all users' interest, which in most cases does not meet an individual's preference.

The MSE results are very valuable because it proves the diversity of our users. It helped us understand the interests of our users are wildly spread, therefore, a recommender system other than popular items and past purchased items is necessarily desirable.

- Hit Rate:



We define our Hit rate as the following:

$$\textit{Hit rate for an individual user } u = \frac{\begin{array}{c}\textit{Number of the overlapping top } \min(20, \textit{number of interested categories}) \\ \textit{recommended categories} \\ \textit{and top } \min(20, \textit{number of interested categories}) \\ \textit{most interested categories}\end{array}}{\min(20, \textit{number of interested categories})}$$

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

The result of our Hit Rate graph is very interesting. It is quite obvious that there is a clear diversion as the modified PPC weight changes.
When PPC ranges from 0.1 to 0.6, our hit rate stabilized around 0.841 after the number of neighbors reached 1. This means that with PPC in this range, no matter how many more similar users we introduce, it won't make any further contribution to improve our Hit Rate. This makes perfect sense because when our user data is sparse, introducing more similar users won't help increasing the interest score on a certain category between two users.

When PCC takes on a higher weight – such as 0.7, 0.8, 0.9, our hit rate monotonically increases and converges when the number of similar users reaches 80. This means a couple of things. For one, we initially introduced Jaccard to fix the potential issues that the modified PPC may overlook. When our PPC takes on a higher weight, Jaccard was indeed able to fix the problem that when PPC is 1, it misrepresents how similar two users' interests are.
We can also see that PPC is a good metric to measure user similarities. From the graph above, hit rate increases as long as we introduced one similar user despite the minor fluctuation. This is consistent with what we found in our exploratory data analysis that when most of our users have limited numbers of purchases, they typically don't have strong past shopping preferences. Therefore, positive influence occurred when we introduced the purchase interest of their similar users.
One detail that we need to address is that there are minor fluctuations in our results. These small inconsistencies reflect that some of our users do have strong purchasing preference. Therefore, recommending based on their similar users is not the best strategy for this type of users. Therefore, we segregated this group of users (type B users) and applied a different model (Time SVD) for them.

<u>Integration:</u>

In the final stage, we integrated our two models as one comprehensive business solution. Instead of merging them as one, we decided to apply different models to users at different maturity level. There are three main reasons for this decision:

1. Our two models are set up fundamentally differently from the designing stage. From the very beginning, the user-item model didn't incorporate the concept of time at all. It focused on the user's shopping behaviors and the behaviors of the similar users, while the latent factor model was built with a further interest to leverage data over time. In this way, the user-item model is naturally more useful to produce short-term and

immediate recommendation, while the latent factor model with time stamp is ideal for long term and future targeting recommendation.

2. Further, as users move from one stage to another, for example, from mere browsers to new first-time buyer, from inactive buyers to regular shoppers, from regular shoppers to loyal customers, our recommendation strategy should capture this change and evolve as the users' maturity change. As our customers grow older, their interest and needs would change too. Therefore, applying different recommendations at different stages would be a more appropriate adaptation.
3. Finally, to increase the overall sales for E-commerce businesses cannot solely rely on the website recommendations. Developing consumer interests also plays an indispensable role in increasing customer activeness, stickiness and loyalty. Through carefully curated content recommendations in the format of subscription service, emails, and blogs, companies can build trust and likability amongst consumers. Hence, by applying latent factor model with time stamps to these types of recommendations, we are able to cultivate and develop customer interests, uncovering more sales opportunities. Once a customer goes to the website, the user-item model on the website would give the customer the immediate and most relevant recommendation. We are able to maximize the opportunity to unleash our customers' purchasing potential.

Below is specifically how we decided to implement:

User type	**Purchase**	**View/add-to-cart**	**Time Stamp**	**Model**	**Recommendation format**
A	3 and more	3 and more	Within a year	User Behavior & Neighborhood Hybrid Model	Website /mobile app
B	3 and more	3 and more	over a year	Latent Factor with time SVD	Email subscription, blog, online community
C	1-3	some	n/a	Popular items within the same categories as the items they showed interests	Website, Email subscription, blog, online community
D	0	No activities	n/a	Top popular items across all categories	Website, Email subscription, blog, online community

type A) Users who have at least three purchase transactions and three views or add-to-cart activities, but has a purchase time scope within a year

type B) Users who have at least three purchase transactions and three views or add to cart activities, but has a purchase time scope over a year

type C) Users who do not have any purchase transactions

type D) The rest of the users

For type A users, our model outperformed baseline model and were able to recommend items of high-related interests.
For type B users, the test results proved our hypothesis that time SVD would perform better than SVD, which indicated that users' purchasing behaviors and interests in different categories would indeed change through the time.
For type C and type D users, since we don't have any access to the future data to verify the recommendation quality, we are yet to provide prediction accuracy.

Even though there is yet any ways to accurately test the recommendation quality for Type B, C and D users due to the fact that we only have data within a year, we believe a good recommendation system should be a comprehensive solution that can cater to different needs of users at different maturity.