Project 3 Collaborative Filtering

Zhang Zhicheng 104946990
Zhou Zhengtao 005036433
Hao Shizhong 605035020
Wang Yucong 305036163

Introduction

In this project, we use different methods to build recommendation system on the ratings of movies. The basic idea under the recommendation system is that we use user-item relation, which refers to collaborative filtering, to infer their interests. In the following section, we apply Neighborhood-based, Model-based and Naive three different collaborative filtering methods to build recommendation system and compare their performance.

MovieLens dataset

MovieLens dataset contains user-movie interaction which is ratings, before we move on to collaborative filtering, we first analyze the property of this dataset.

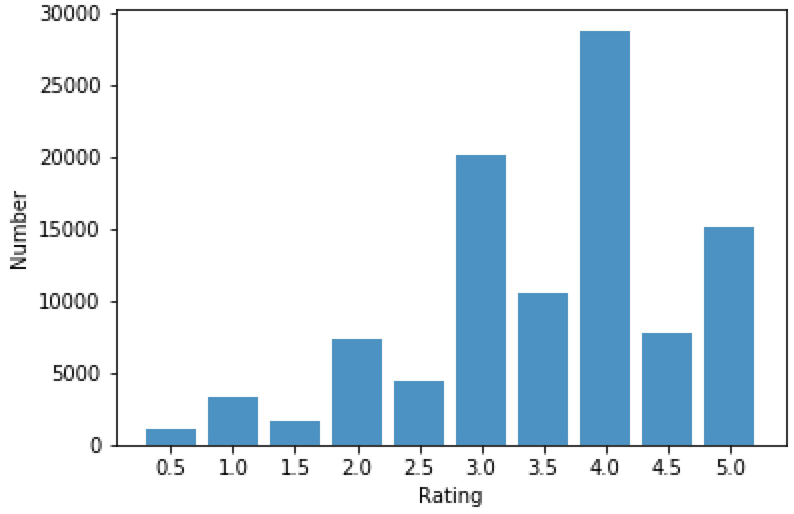
Question 1

The number of available ratings is 100004, and the number of possible ratings is calculated by the number of users times the number of movies, which is 6083286. The sparsity is

$$Sparsity = \frac{Total\ number\ of\ available\ ratings}{Total\ number\ of\ possible\ ratings} = 0.016439141608663475$$

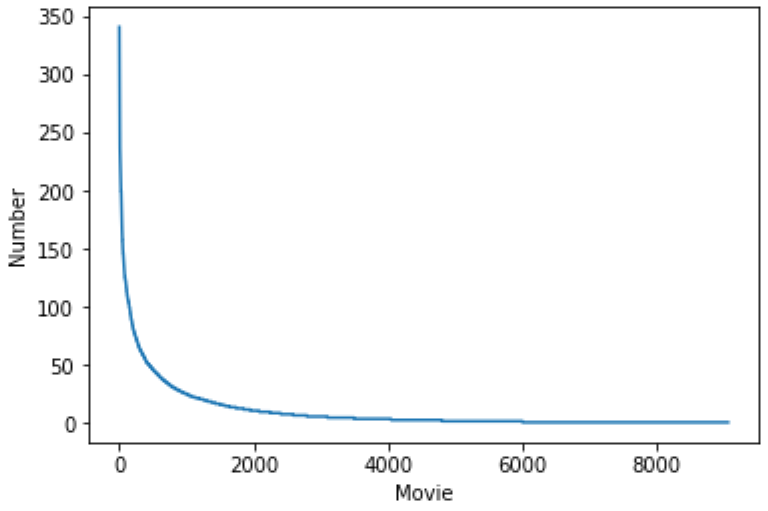
Question 2

The frequency of the rating values is shown below



Question 3

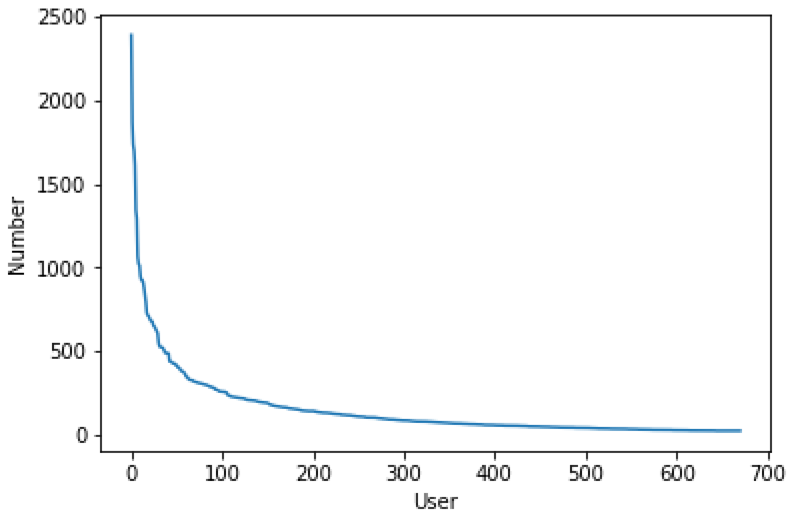
The distribution of ratings among movies is shown below



From the figure above, we can see the curve decrease dramatically, which means some popular movies get much more ratings than those unpopular ones and most movies receive few ratings .

Question 4

The distribution of ratings among users is shown below



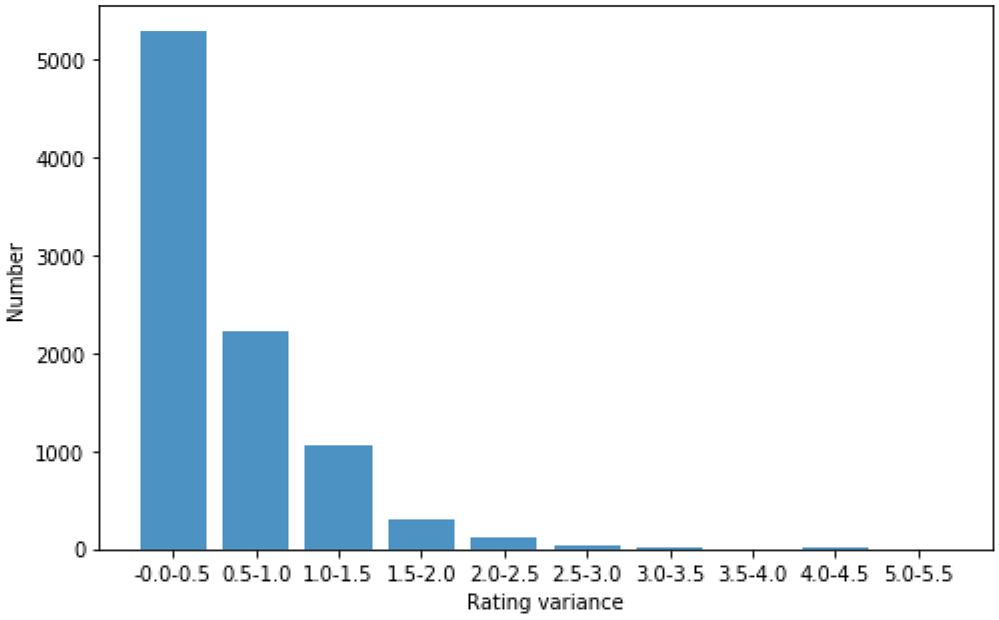
The distribution ratings among users and movies are roughly the same, which means most users only viewed few movies.

Question 5

A large number of movies receive few ratings, and a few of movies receive much more ratings. It can be explained by only a few of popular movies get more attention which receive more ratings but most movies are unpopular which only a few of people viewed and rated. Such salient feature makes it harder to train the model and the model has less accuracy and may not be stable because the difference between popular and unpopular movies, especially for unpopular movies having few ratings.

Question 6

The variance of the rating value received by each movie is shown below



Most movies has little variance of ratings, only few movies has large variance of ratings. It can deduce that most people have the same feeling on a movie and gave roughly the same ratings.

Neighborhood-based collaborative filtering

Question 7

The formula for μ_u in terms of I_u and r_{uk} is

$$\mu_u = \frac{\sum_{k \varepsilon I_u} r_{uk}}{|I_u|}$$

Question 8

Based on the definition of I_u , $I_u \cap I_v$ means set of item indices for which ratings have been specified by both user u and user v.

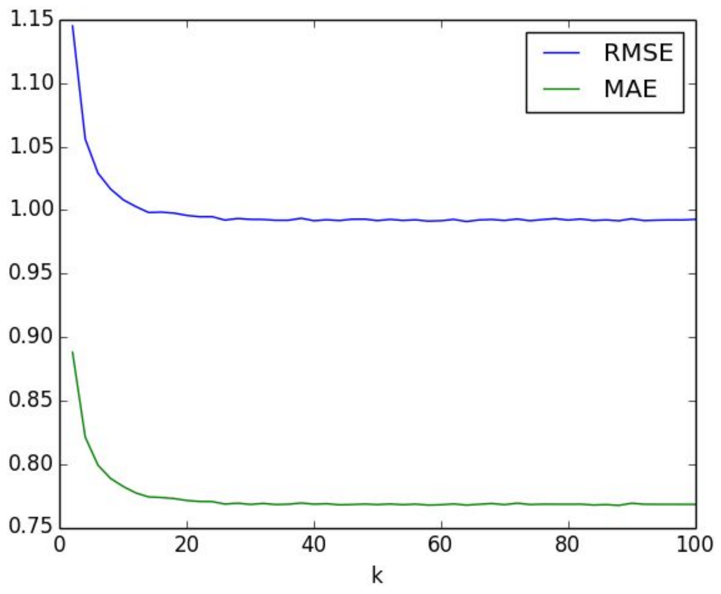
$I_u \cap I_v = 0$ is possible, which means user u and user v do not have the same item rated. Since the rating matrix is sparse, users only viewed a few movies, it cannot guarantee that users u and v must have rated the same movie, especially users u and v only rated few movies.

Question 9

The reason behind mean-centering the raw ratings $(r_{uk} - \mu_u)$ in the prediction function is to decrease the difference in absolute Euclidean distance among users. For example, user u tend to rate all item highly have higher rating score on an item if he likes it, but user v tend to rate all item poorly have lower rating score even if he has the same feeling as user u. Therefore, r_{uk} can not represent the taste of users on an item, but in the prediction function $(r_{uk} - \mu_u)$ can tell the similarity among users.

Question 10

In this part, we designed a K-NN collaborative filter using KNNWithMeans from Surprise package. The k value is swept from 2 to 100 with step of 2. The similarity measurement is set to be 'Pearson'. Then 10-fold cross-validation is performed, and average RMSE and MAE are plot over each k. Below is the plot.



Question 11

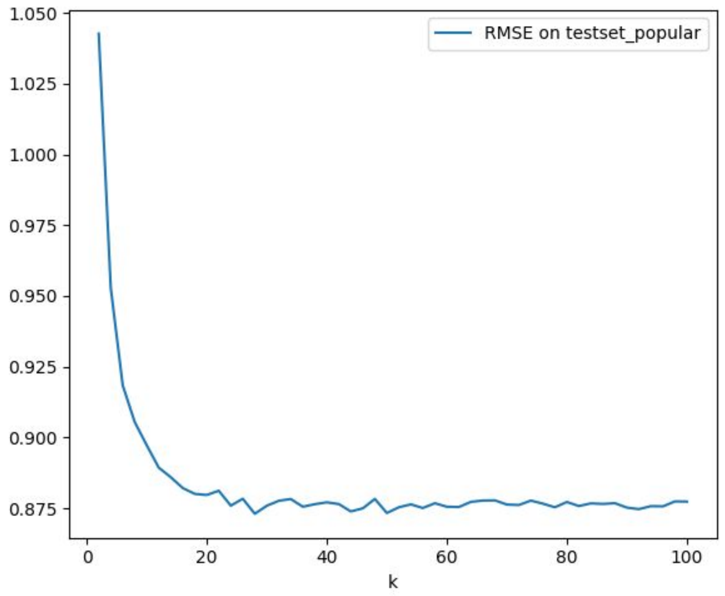
From the plot above, it's easy to see that overall RMSE and MAE are monotonous decreasing as k increases. This phenomenon means that the performance of K-NN collaborative filter is better when taking more neighbors into account. However, after k passes around 30, the decreasing of RMSE and MAR becomes trivial. Therefore, the ‘minimum k’ is **30**, and the steady state value of RMSE is about **0.99**; the steady state value of MAE is about **0.77**.

Question 12

In this part of the project, we analyzed the performance of the k-NN collaborative filter in predicting the ratings of the movies in the trimmed test set. We considered the following trimming:

• Popular movie trimming
• Unpopular movie trimming
• High variance movie trimming.

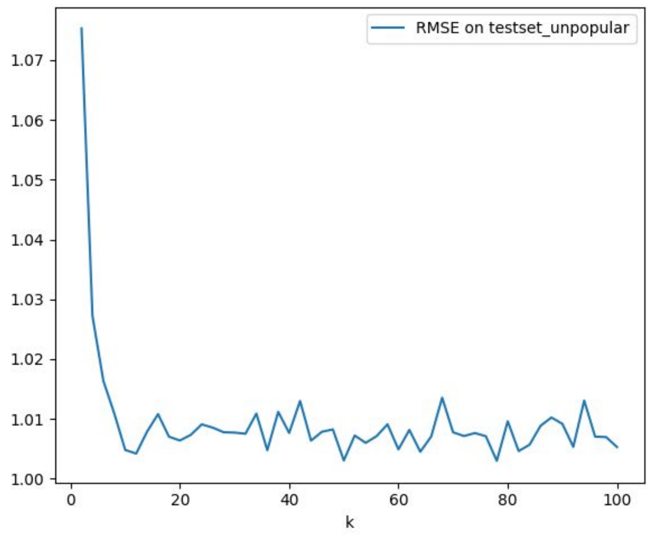
The dataset was partitioned into 10 equal sized subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the filter, and the remaining 9 subsets are used to train the filter. The cross-validation process is then repeated 10 times, with each of the 10-subsets used exactly once as the validation data.
In the popular test set which only contains movies received more than 2 ratings, we sweep number of neighbors to predict the performance of KNN filter. The k value is swept from 2 to 100 with step of 2, and for each k we computed the average RMSE across all 10 folds. The plot is shown below, and the min_RMSE is **0.872405** on popular testsets.



From the figure we can see that the error rate goes down when k is increasing, also the curve becomes stable after RMSE reached the minimum point. So the filter is quite reliable to predict the ratings for movies in trimmed popular test set which satisfies our expectation.

Question 13

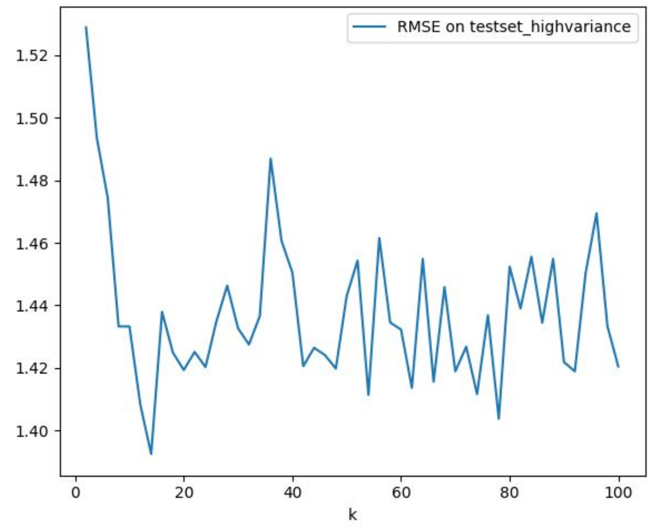
For unpopular testset the plot is shown below, and the min_RMSE is **1.00272** on unpopular testset.



From the figure we can see that the error rate also goes down when k is increasing, the curve becomes relatively not stable after RMSE reached the minimum point comparing to the predicting in popular sets. The filter is basically reliable to predict the ratings for movies in trimmed unpopular test set. The reason might be that unpopular movies could have different judge standards for various of users, unlikely, for popular movies people might have similar taste on the definition of popular. We can draw the conclusion that when comparing to unpopular trimming sets, the KNN filter performs well on popular sets.

Question 14

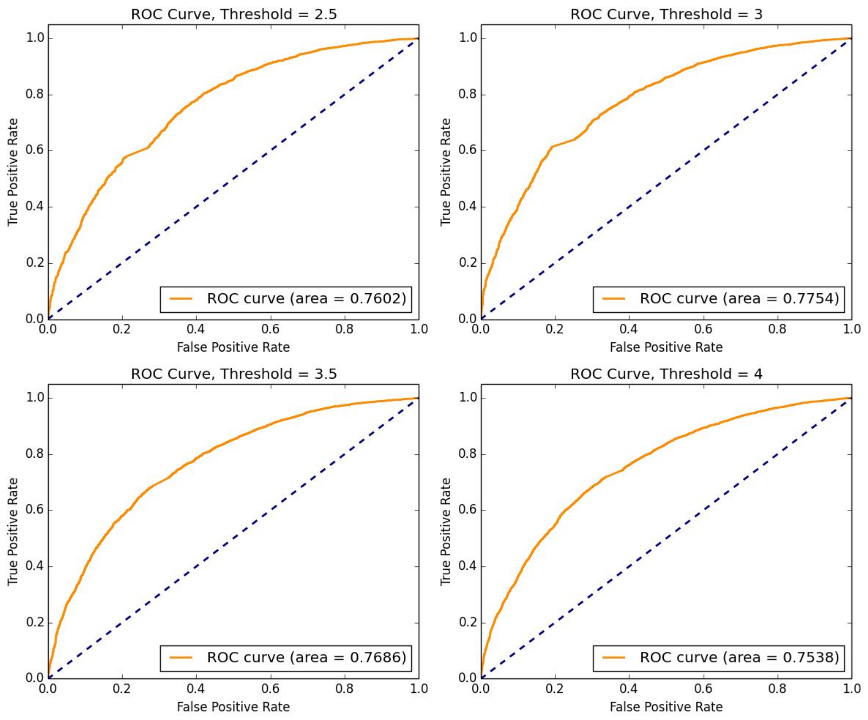
For high variance testset the plot is shown below, and the min_RMSE is **1.39232** on high variance testset.



From the figure we can observe that the error rates are always jumping as the value of k is increased, the curve doesn't have a specific tendency when comparing to the prediction in popular and unpopular sets. We could say the filter is not quite reliable to predict the ratings for movies in trimmed high variance test set.
The reason might be that the trimming method for high variance sets is designed with randomness, we trim the test set to contain movies that has variance of the rating values received of at least 2 and has received at least 5 ratings in the entire dataset. Thus, the test set is small and reduced universality. When comparing with the other two trimming methods, the K-NN filter performs bad on high variance sets, and the error rate cannot maintain stability.

Question 15

In this part, the ROC curve of K-NN collaborative filter is plotted to evaluate its performance. The k-value is 30, which is the optimum value found in previous section. Since ROC curve is used to evaluate the performance of binary classifier, to extend it to collaborative filter, we need to binarize the ground-truth ratings into 'like' and 'dislike'. This process is done by comparing the rating with a threshold. If the rating is less than the threshold, it is labeled '0'. Otherwise it is labeled '1'. In this section, we also used four different thresholds: [2.5, 3, 3.5, 4]. Below is the result.



From the result, we can see that is the ROC curves of K-NN collaborative filter look like arcs, which is far from the ideal shape. This implies collaborative filtering is less accurate than supervised learning. It can also be seen that changing threshold does not impact its performance significantly. However, the area under ROC curve is maximized when the threshold is 3 (there is some random variables in KFold, so sometimes it maximized at 3.5). Intuitively, it is best to divide all ratings into 'like' and 'dislike' by an intermediate value rather than some extreme values.

Threshold	2.5	3	3.5	4
AUC	0.7602	0.7754	0.7686	0.7538

Question 16

The unconstrained optimization problem for matrix factorization is given in the above optimization problem, U and V are matrices of dimension m × k, and n × k respectively, where k is the number of latent factors. However, in the above setting it is assumed that all the entries of the rating matrix R is known, which is not the case with sparse rating matrices. Fortunately, latent factor model can still find the matrices U and V even when the rating matrix R is sparse. It does it by modifying the cost function to take only known rating values into account. This modification is achieved by defining a weight matrix W in the following manner:

$$W_{ij} = 1 \textit{ when } r_{ij} \textit{ is known};$$
$$W_{ij} = 0 \textit{ when } r_{ij} \textit{ is unknown}.$$

So we can formulate the optimization problem as equation 5.

$$\textit{minimize}_{U,V} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2$$

For U fixed, it could be treated as a least-squares problem.
The function to minimise is

$$C = \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2$$

Differentiating with respect to vector V^T results in:

$$\begin{aligned}\frac{\partial C}{\partial V^T} &= \frac{\partial W(R-UV^T)^2}{\partial V^T} \\ &= \frac{\partial W(\|R\| - 2RUV^T + \|UV^T\|^2)}{\partial V^T} \\ &= -2RU + 2U^T V U\end{aligned}$$

since $W = 1 \textit{ when } R \textit{ is known}$

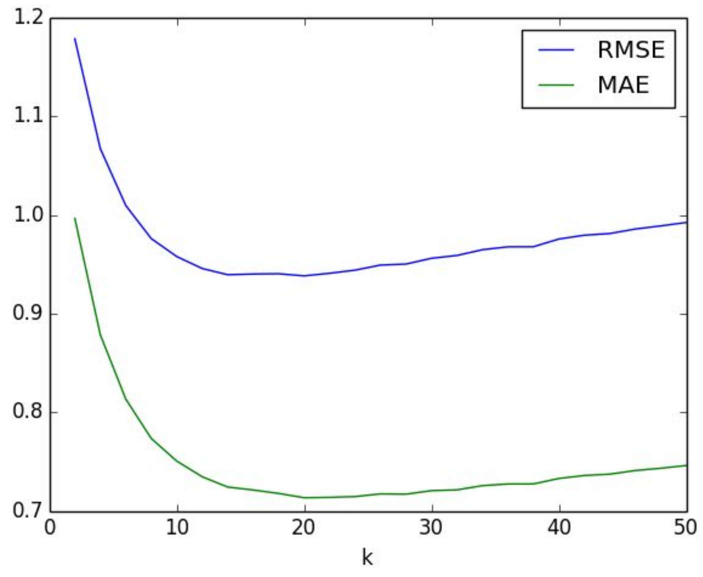
And the second derivative is

$$\frac{\partial^2 C}{\partial V \partial V^T} = 2U^T U \geq 0$$

which is a positive semi-definite matrix. Therefore, the optimal problem is a convex function.

Question 17

In this section, we use non-negative matrix factorization to perform the collaborative filtering. The k value is swept from 2 to 50 with step of 2. Then 10-fold cross-validation is performed, and average RMSE and MAR are plot over each k. The plot is shown below.



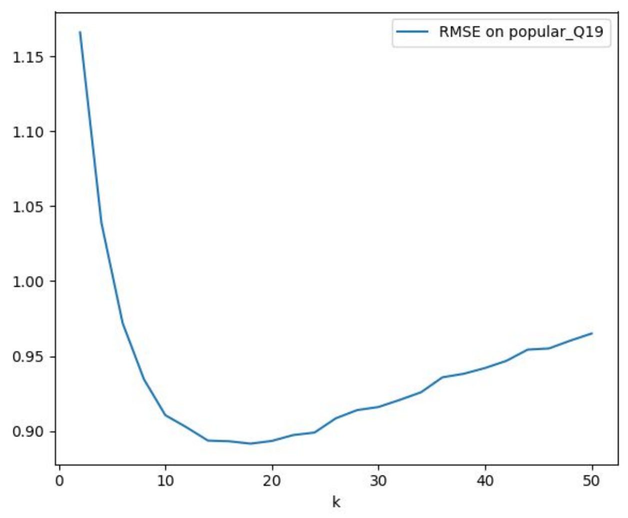
Unlike the K-NN case, the prediction errors do not monotonous decrease as k increase. Obviously, when k is very small, there is not enough information to prediction the rating, so the error rate is high. However, when k is very large, the error rate also increases, which means having too much information in NMF matrix also harms the prediction accuracy.

Question 18

Overall, the RMSE and MAE are minimized when k = **20**, and the minimum average RMSE is about **0.9383**; the minimum average MAE is about **0.7133**. These values are smaller than the K-NN case, which means using NNMF in MovieLens dataset does yield better result than K-NN collaborative filtering. Since there are 19 genres in the dataset (including 'no genres listed'), the optimum k value is not exactly same as the number of genres, but very close. It is very amazing to see that human-defined movie genres can be observed using pure mathematical methods.

Question 19

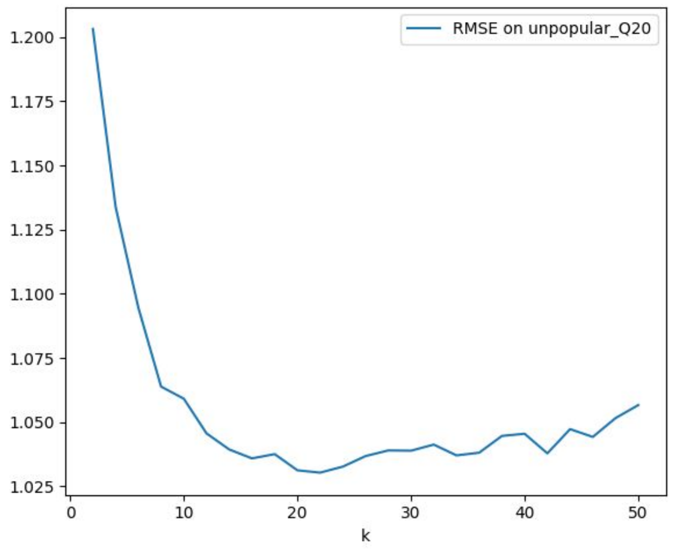
Since we have designed the NNMF filter in the previous questions, now we will test the performance of the filter in predicting the ratings of the movies in the trimmed test set. We used the same trimming operations as before.
The average RMSE against k is shown below, and min_RMSE for popular testset is **0.889776**.



From the figure we can see that the error rate goes down then goes slightly up when the number of neighbors is increasing, the curve becomes relatively stable when RMSE reached the minimum point. The filter is basically reliable to predict the ratings for movies in trimmed popular test set. There exists a best value k at which point the NNMF filter performs best on popular test set.
The reason might be that popular movies could have common judge standards for a specific number of user neighbors who contribute to the ratings of movies, we can draw the conclusion that the NNMF filter performs well on popular sets, and it works best on a specific amount of neighbors which should be considered when building a recommender system.

Question 20

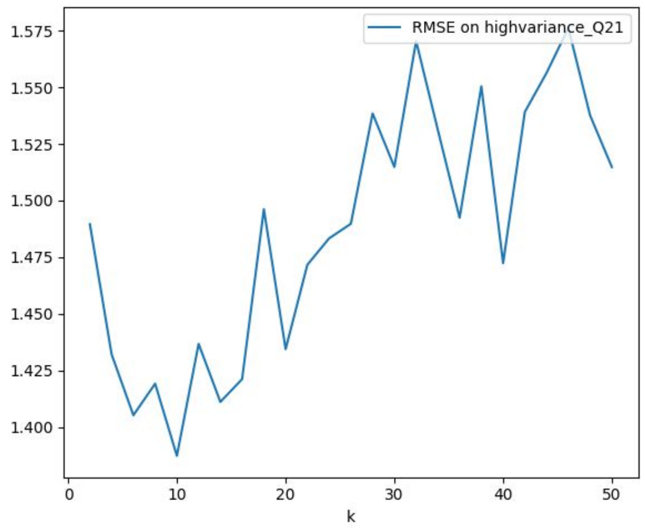
The average RMSE against k is shown below, and min_RMSE for unpopular testset is **1.02971**.



From the figure we can see that the error rate goes down and keeps steady when the number of neighbors is increasing, the curve becomes relatively stable when RMSE reached the minimum point. The filter is basically reliable to predict the ratings for movies in trimmed unpopular test set. The NNMF filter performs well on unpopular sets.

Question 21

The average RMSE against k is shown below, and min_RMSE for high variance testset is **1.38701**.

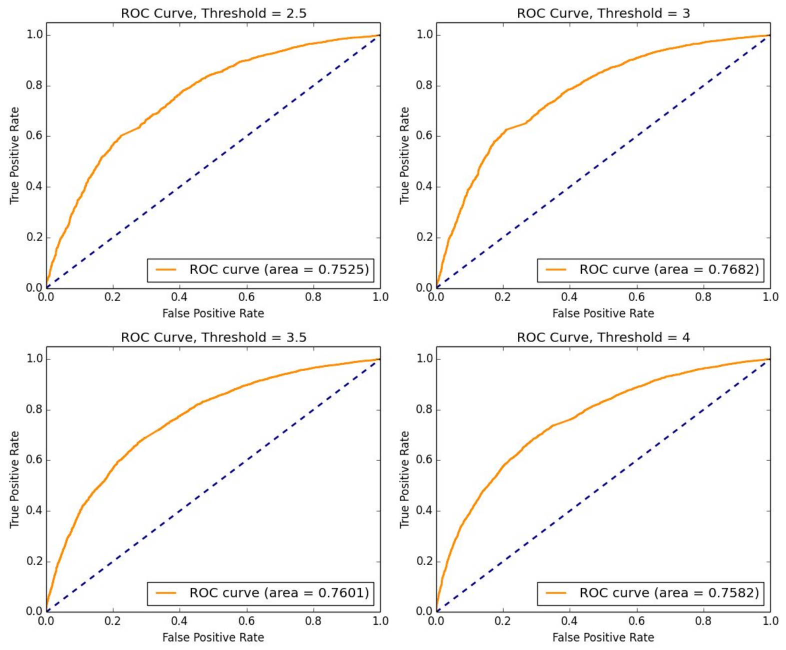


From the figure we can observe that the error rates are always jumping as the value of k is increased, the curve doesn't have a specific tendency when comparing to the prediction in popular and unpopular sets. We could say the NNMF filter is not quite

reliable to predict the ratings for movies in trimmed high variance test set.
Because the trimming method for high variance sets is designed with randomness, we trim the test set to contain movies that has variance of the rating values received of at least 2 and has received at least 5 ratings in the entire dataset. Thus, the test set is been cut-off and becomes small and reduced universality. When comparing with the other two trimming methods, the NNMF filter has a poor performance on high variance sets, and the error rate cannot maintain stability.

Question 22

In this part, the ROC curve of NNMF-based collaborative filter is plotted to evaluate its performance. The k-value used is 20, which is the optimum value found in previous section. Again, the ground-truth ratings are binarized with different thresholds so we can perform binary comparison. The results are shown below.



Same as the K-NN case, the area under ROC is maximized when threshold is 3, and the areas are slightly smaller for other thresholds.

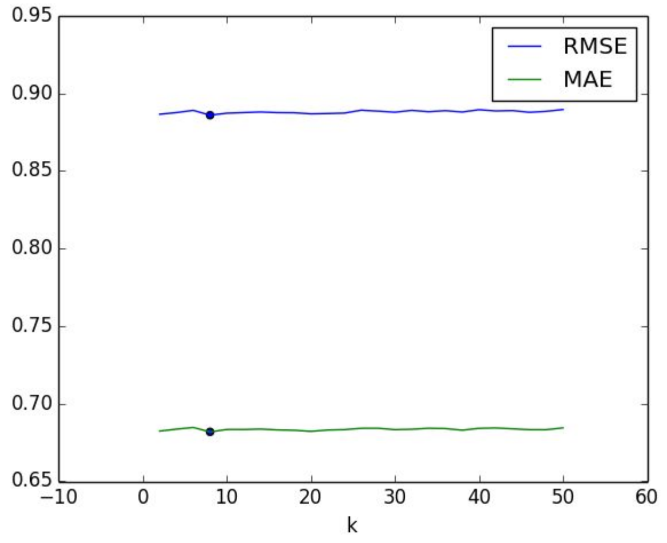
Threshold	2.5	3	3.5	4
AUC	0.7525	0.7682	0.7601	0.7582

Question 23
In this section, we want to exam the relation between latent factors and movie genres. To do that, we first search for movies with the top ten greatest factor values within the column picked, then translate their index into raw movie ID. After that, print the corresponding genres by looking up ‘movies.cvs’. Since the result of NMF is not same every time, we sampled these genres of top ten movies in a single run. The result is listed below.

Column #2	Column #5	Column #7	Column #11
['Comedy|Drama']	['Drama']	['Action']	['Drama|Mystery|Sci-Fi']
['Action|Adventure|Drama']	['Comedy|Crime']	['Comedy|Mystery']	['Horror|Sci-Fi']
['Action|Adventure|Sci-Fi']	['Comedy|Crime|Drama']	['Comedy|Drama']	['Animation|Sci-Fi']
['Action|Sci-Fi|Thriller']	['Drama']	['Children|Comedy']	['Comedy']
['Action|Drama|Sci-Fi|Thriller']	['Action|Comedy']	['Action|Sci-Fi|Thriller|Western']	['Drama|Horror|Mystery|Thriller']
['Action|Sci-Fi|Thriller|IMAX']	['Drama|Mystery|Sci-Fi']	['Comedy']	['Adventure|Drama|Fantasy|Romance']
['Children|Comedy']	['Drama|Romance|War']	['Documentary']	['Action|Adventure|Sci-Fi|IMAX']
['Adventure|Animation']	['Drama']	['Drama']	['Action|Comedy|Drama']
['Action|Adventure|Fantasy|Romance|IMAX']	['Drama']	['Horror|Sci-Fi']	['Action|Thriller']
['Comedy']	['Drama|Horror|Mystery']	['Comedy|Drama']	['Drama|Romance']

Among the top ten genres of these latent factors, some are pretty homogeneous, while some are not. First let’s take a look at column #2, the majority in it contains ‘Action’, and ‘Adventure’ also appeared multiple times, so this latent factor is like to be a composed genre of ‘Action’ and ‘Adventure’. Take column #5 as an example, major genres in it are ‘Drama’, so this latent vector is very likely to be corresponding to the genre ‘Drama’. The column #7 is not that pure. Although half of them contain ‘Comedy’, but the rest of them do not have obvious pattern. Column #11 is similar to this case, while ‘Sci-Fi’ and ‘Action’ appear multiple times, but the rest of them do not share much similarity. In conclusion, the latent factors found by NNMF is correlated with genres defined by human, but they are not one-to-one corresponded. NMF tries to find the similar patterns between different ratings, and the way users rate movies are differed by genres. This characteristic makes NNMF more interpretable than other collaborating filters.

Question 24
In this section, MF with bias is performed to factorize the raw sparse data matrix. The algorithm used is SVD. The k value is swept from 2 to 50 with step of 2. Then 10-fold cross-validation is performed, and average RMSE and MAR are plot over each k. The plot is shown below.



At first glance, it is hard to tell where the minimum points of RMSE and MAE are, since their values do not change with k significantly. Even worse, since K-fold introduced some random variables, the minimum points do not settle on the same k value every time. We added python functions to point out the minimum points so they can be easily identified.

Question 25

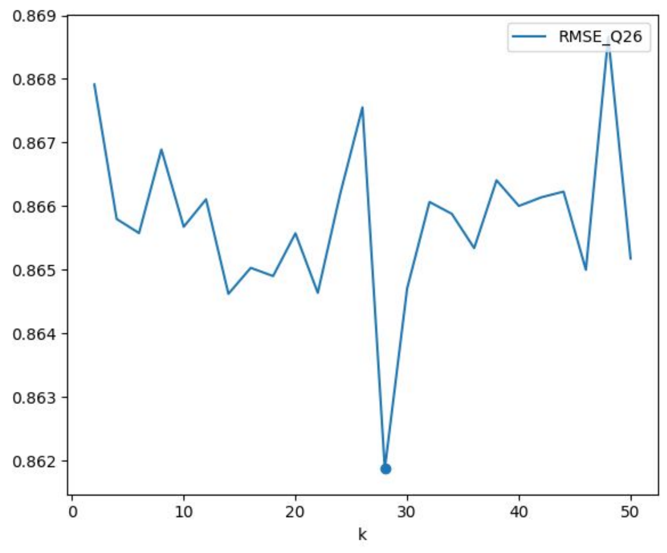
After multiple runs, we found the minimum points of RMSE and MAE usually happen when k is **8**. In this case, the minimum average RMSE is about **0.8860**, and the minimum average MAE is about **0.6819**. Each value is significantly smaller than previous cases, which means MF with bias has a better performance on collaborative filtering.

Question 26

Since we have designed the MF with bias filter in the previous part, we test the performance of the filter in predicting the ratings of the movies in the trimmed test set. We used the same trimming operations as before.

We plot average RMSE against k to evaluate its performance using 10-fold cross validation. K_value is swept from 2 to 50 in step sizes of 2.

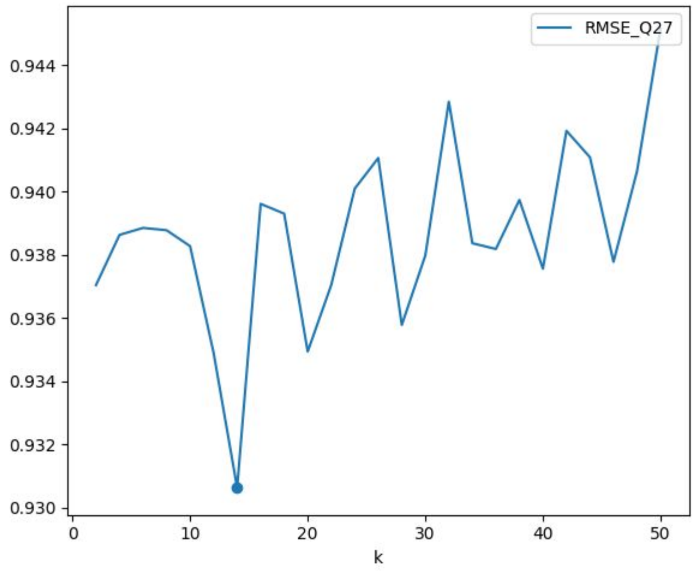
The plot is shown below, and min_RMSE for popular testset is **0.8618772649496844**, when k is **28**.



From the figure we can observe that the error rates are always jumping randomly as the value of k is increased, which is worse, the curve doesn't have a specific tendency. The error curves on former filters are smoother than on MF with bias. We could say the Matrix factorization with bias filter is not quite reliable to predict the ratings for movies in trimmed popular test set.

Question 27

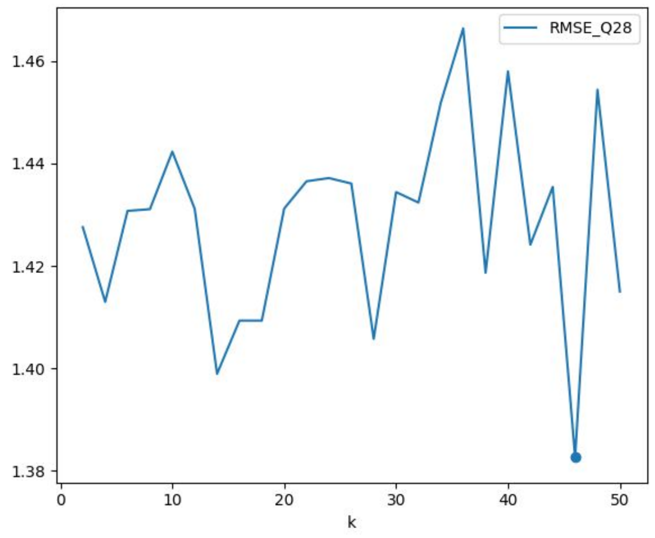
The plot for unpopular testset is shown below, and min_RMSE for unpopular testset is **0.9306250671666769**, when k is **14.**



We can also see that the error rates are always goes down and up again as the value of k is increased, the curve doesn't have a specific tendency. The error curves on former filters are smoother than on MF with bias. We could draw the conclusion that the Matrix factorization with bias filter is not quite reliable to predict the ratings for movies in trimmed unpopular test set.

Question 28

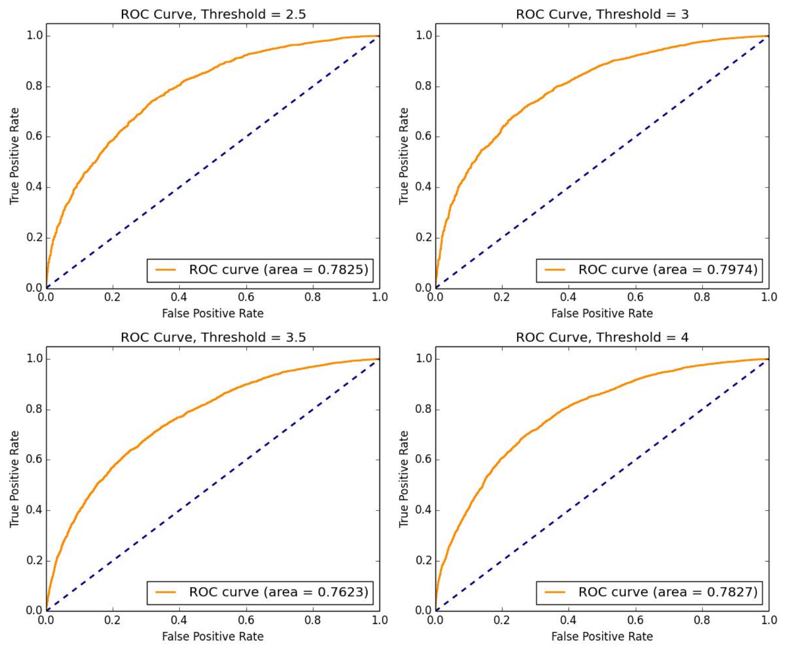
The plot for high variance testset is shown below, and min_RMSE for high variance testset is **1.382642646927811**, when k is **46**.



For the high variance test set, the RMSE curve is not steady as the value of k is increased, and it doesn't have a specific tendency.
Since the error curves on former filters are smoother than on MF with bias. We could draw the conclusion that the Matrix factorization with bias filter shows a poor performance on prediction of the ratings for movies in trimmed test sets.

Question 29

Similar to previous sections, the ROC of MF with bias is plotted using different thresholds. The k value is set to be 8, which is the optimal number of latent factors found in Question 25. The results are shown below.



Threshold	2.5	3	3.5	4
AUC	0.7825	0.7974	0.7623	0.7827

Same as the K-NN and NNMF cases, the area under ROC attains maximum when the threshold is 3. Changing the threshold does not affect the ROC significantly. Although the area under the curve when threshold is 3.5 is less than that when threshold is 4, but this can be the result of random noise.

Question 30

In this part, we designed a naive collaborative filter which simply predicts ratings by returning the mean rating of a user. Since this filter is not a built-in filter in surprise package, we built our own version using AlgoBase class. When estimate a rating of movie *i* by user *u*, it just returns the mean value of all ratings by user *u*, including these in test set. This filter totally ignores the information from individual movies. Then its performance is evaluated using 10-fold cross validation. In fact, whether use cross-validation does not make any difference here, since this filter does not take any information from the train set.

The average RMSE is about **0.9554**. Surprisingly, this score really isn't bad for such a simple algorithm. In fact, it is fairly close to methods mentioned in previous section. This result reveals that each user tends to rate movies with a mean value. They would only rate extremely when they really like or really dislike one movie. But this does not mean users are irrational, if the movies they very like or dislike only counts a small portion of all the movies, such low error rate is reasonable, and this does not mean this naive filter is useful in collaborative filtering.

Question 31

Since we have designed the naive collaborative filter in the previous section, we test the performance of the filter in predicting the ratings of the movies in the trimmed test set. An important thing to note about the naive collaborative filter is that there is no notion of training.
We split the dataset into 10 pairs of train set and test set and for each pair predict the ratings of the movies in the test set using the prediction function (no model fitting required). Then compute the RMSE for this fold and repeat the procedure for all the 10 folds. The average RMSE is computed by averaging the RMSE across all the 10 folds.
We applied naive filter on the three trimmed test sets, and we computed the average RMSE by averaging the RMSE across all 10 folds. For popular testset we got the result that average RMSE is **0.93891711206**.

Question 32

We computed the average RMSE by predicting the ratings of movies for unpopular testset. For unpopular testset the average RMSE is **0.990234564232**.

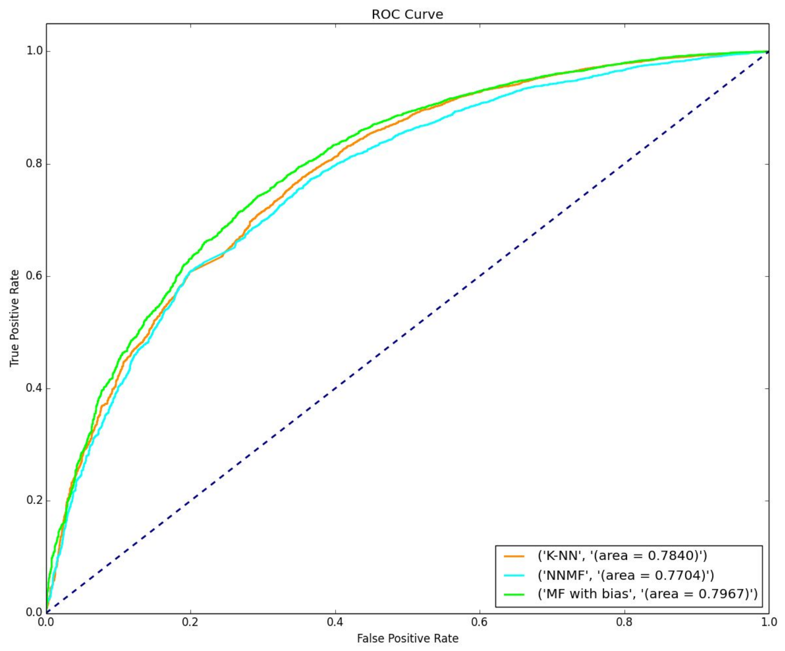
Question 33

We computed the average RMSE by predicting the ratings of movies for high variance testset. For high_variance testset the average RMSE is **1.46775993671**.

Comparing the three results on different trimmed sets, we could observe that naïve collaborative filter shows its best performance on popular test set and for high variance set it performs not as well as the other two, which is consistent with our previous understanding on the other filters.

Question 34

In this part, we plot the ROC curves for K-NN, NNMF, and MF with bias based collaborative filters to compare their performances. Below is the plot.



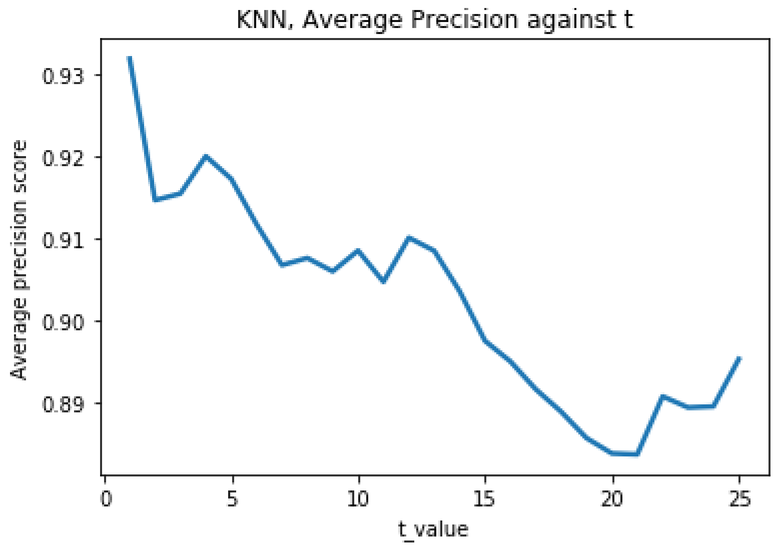
It can be seen that while there is no big difference between all three curves, a good collaborative can outperform a bad one in almost all regions. The plot shows MF with bias based filter has the best perform (area under curve is 0.7967), followed by K-NN (area under curve is 0.7840). NNMF based collaborative filter is the last (area under curve is 0.7704). Based on this observation, when design a new collaborative filter, MF with bias method should be preferred since it is most likely to have the best performance.

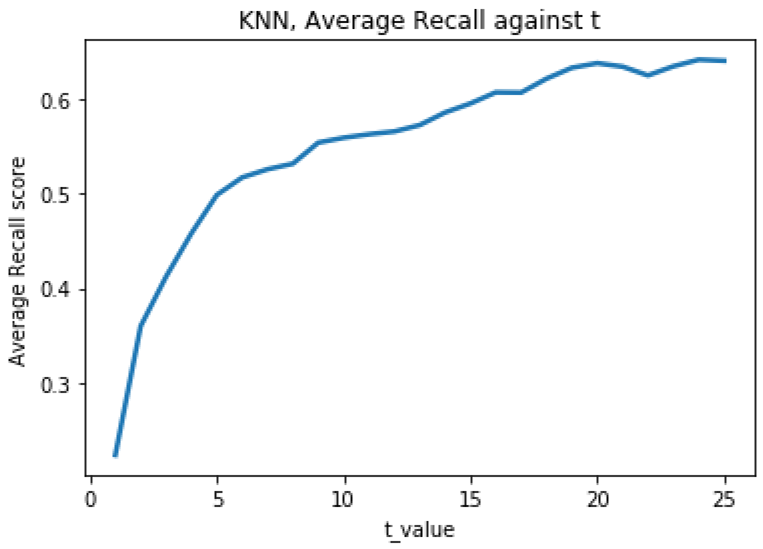
Question 35

According to the equation 12 and 13 given in the question, we can see that the precision here is the ratio of the intersection of the recommended set and the set liked by user and the recommended list itself. That means the precision is the percentage of right commendation made according to user's actual preference in the recommended list. On the other hand, recall function is ratio of the intersection of the recommended set and the set liked by user and the set liked by user itself. It represents the percentage of the correct recommendation with the user's preference to the user's actual rating.

Question 36

In this part, we are required to calculate average precision and recall as a function of t (the size of recommended list) and sweep within t value between 1 to 25 in KNN filter. After prediction was made, we build a dictionary according to user id and use the estimated rating and actual rating as tuples. By dropping the user who have no ground positives and less ratings than the value of t, we are able to get an individual precision for each user. Taking average of there precision will then give us the average precision score. It is the same to obtain recall score.

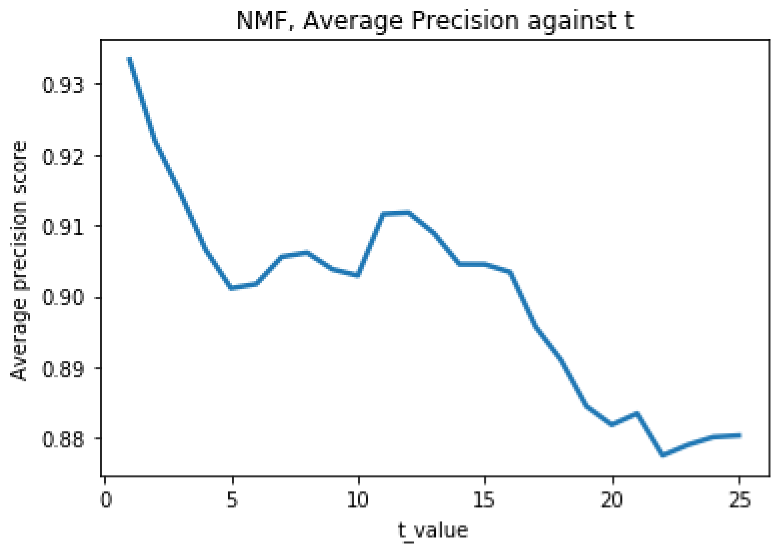


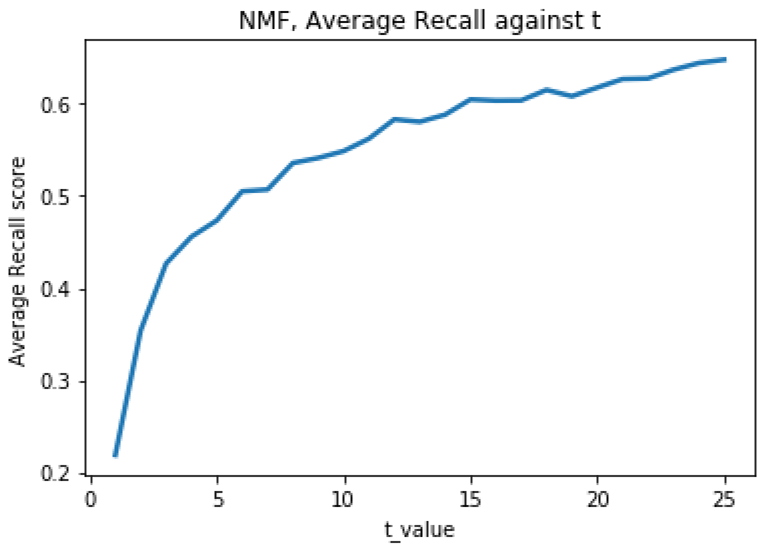


From the plot, we can see that the average precision score decreases with the increase of the size of the recommended list (t value). However, the recall score increases with the size of the recommended list (t value).

Question 37

In this part, we perform the same task over NNMF-based collaborative filter predictions. Here are the plot we obtained:

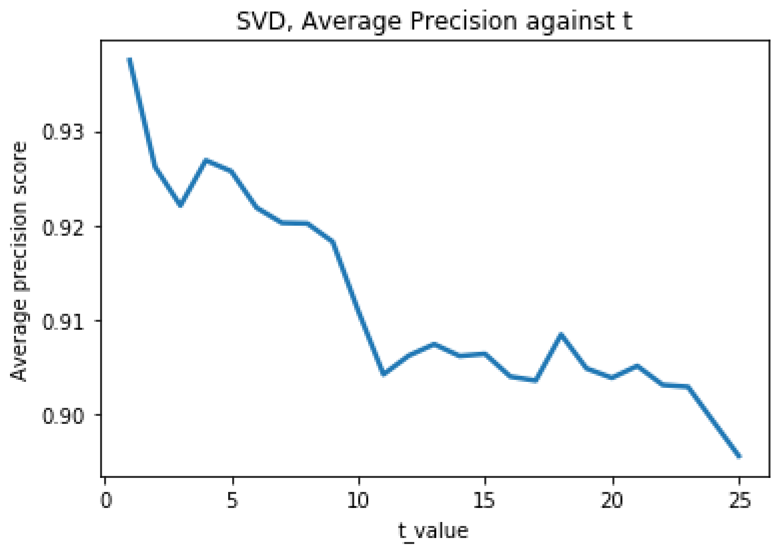


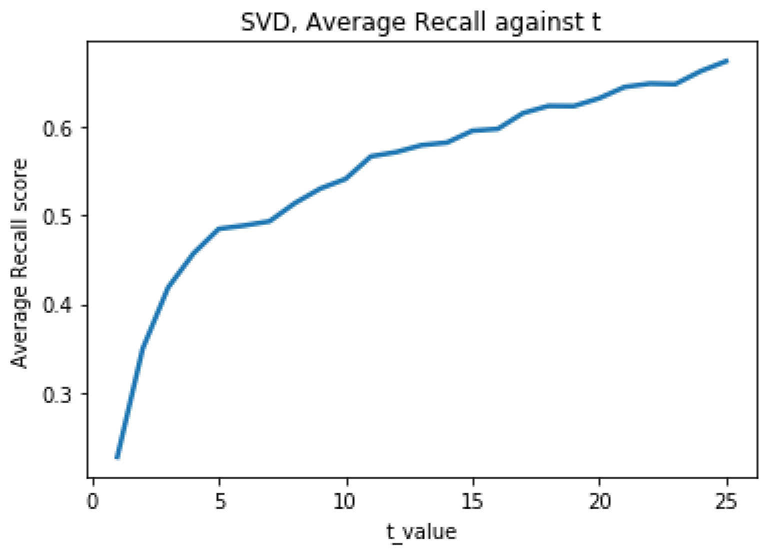


In NNMF-based collaborative filter, we can see that it has the similar shape of average recall score as the KNN filter. However, the average precision has a slight difference. It drops rapidly in the region of t<5 than it increases a little bit from t=5 to t=13. Then it drop again with t increases.

Question 38

In this question, we perform the same task over MF with bias-based collaborative filter predictions.

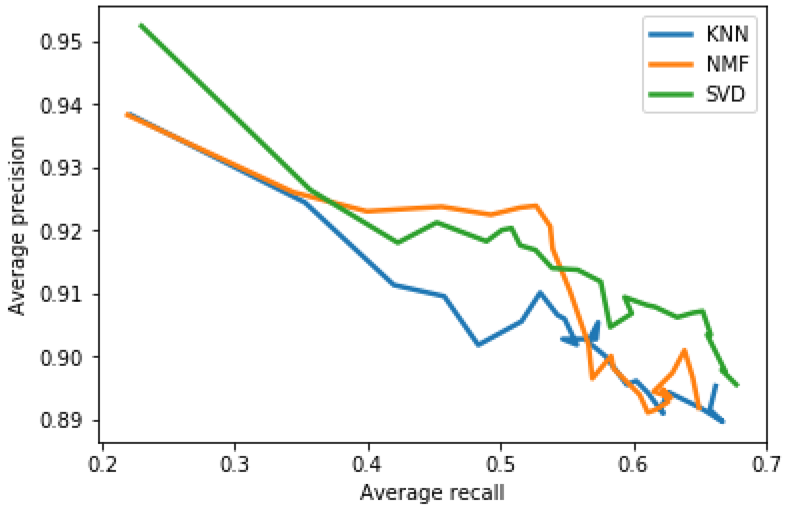




We can see that the average recall have a relatively linear shape after t=10 but is still continuous increasing with bigger t value. The average precision decrease in overall shape but have strong fluctuation at certain region (e.g. t=5, t=12, t=17).

Question 39

Then we plot the precision-recall curve with the results from 36,37,38 in the same figure.



From the graph, we can see that all of the three curve are showed an inverse relationship between average precision and average recall. There is a little fluctuation at the right side of the graph when average precision drop down around 0.92. From this figure, we can see that the overall precision and recall of MF with biase filter is better than the other. While NNMF and KNN are having similar result when t is low. With the increase of the size of recommended list NNMF has better precision at the same recall rate. Therefore, we can conclude that MF with bias is one of the best among these three filter to our recommend system.