Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 3 Report

Collaborative Filtering

ECE 219
Data Mining
February 26, 2021

1 Introduction

Recommender systems have gained popularity over the years. These are systems that recommend selections or choices based on previous inputs by the users. The suggestions made by these systems are curated by an individual's likes and dislikes which increases the chances of the recommendation being accepted by the user. Two important concepts in recommender system are "user" and "item", which are defined as below:

- User: Entity to whom recommendation is made
- Item: Recommendation made to the entity (user)

Recommender systems are used extensively in popular social media sites like Facebook, Twitter and Instagram, as well as streaming platforms like Netflix, Hulu, Amazon Prime etc. They're even used in food and shopping apps these days.

2 Collaborative filtering models

Collaborative Filtering is a technique used by recommender systems. This method involves leveraging multiple data points by many users and using them in a collaborative manner to make suggestions/recommendations. These models are very useful because often datasets are sparse, in that they miss a lot of values. It is important to account for and fill in those missing values for a comprehensive analysis.

Collaborative Filtering is based on the idea that similarity can help make accurate predictions of the missing values. This means, if two users record similar responses for a certain trend in the dataset, then if a particular field of this trend is missing a value from one of these two users, the corresponding data point of the other user can be used to make a prediction about and fill in a missing value. For example, if two users A and B are known to give favourable as well as similar ratings to restaurants that serve continental food, then if one particular restaurant serving continental food is missing a rating from A, an estimation of A's rating for that restaurant can be made from B's rating for that restaurant, reason being that both find this trait favourable and are also known to give similar ratings so there are high chances of this prediction being accurate, as compared to say C, that doesn't give favourable ratings to such restaurants.

Most collaborative filtering methods/models rely either on inter-item correlation or inter-user correlation. In this project, we will only implement user-based collaborative filter model, which is dependent on inter-user correlation.

Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 3 Report

Collaborative Filtering

ECE 219
Data Mining
February 26, 2021

3 MovieLens dataset

QUESTION 1:

Sparsity is defined as the ratio of the available ratings to the possible ratings.

$$\text{Sparsity} = \frac{\text{Number of available rating}}{\text{Number of possible ratings}} \tag{1}$$

The number of available ratings is the number of ratings present in the dataset. The number of possible ratings is the total number of ratings that may be present in the dataset. This is given by the product of total number of users and the total number of movies to be rated.

The Sparsity of the ratings dataset was found out to be as follows:

$$\text{Sparsity} = 0.016999683055613623$$

It can be seen that the rating matrix is quite sparse (sparsity $\approx 1.7\%$). This can be naturally attributed to the fact that there are 9742 movies in the dataset in total and each and every user has a limited capacity of movies that they would watch and rate.

QUESTION 2:

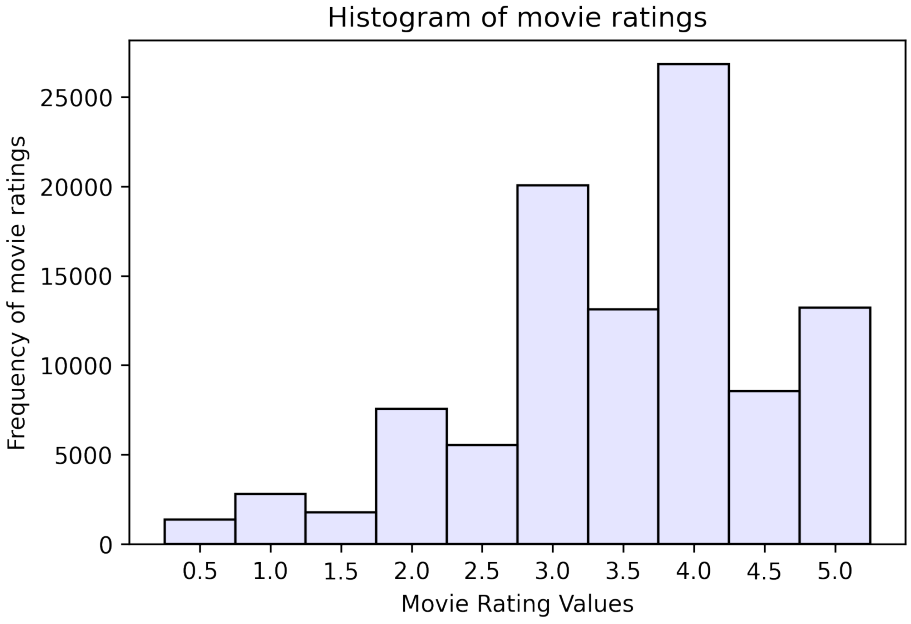

Figure 1: Histogram of movie ratings

The histogram shows that this dataset contains movies popular with the audience as it majorly contains ratings ranging from 3-5 with 3-4 having a major share in that and more movies received ratings in the range of 4-5 than in 0-3. This also shows that people are more likely to rate a movie if they really liked it, and may not even bother about movies they didn't like.

QUESTION 3:

We extract unique movies and then count the number of ratings that each movie received. After that the movies were grouped in a manner where the movies with the first index had the highest number of ratings, then the second one had the second highest number and so on and so forth. The graph is a monotonically decreasing function as shown in Fig. 2.

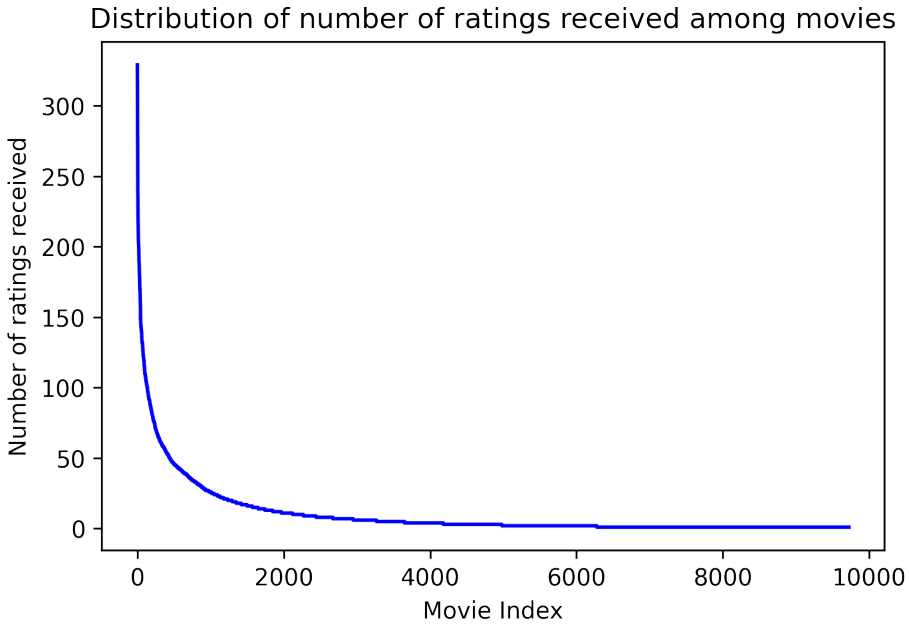

Figure 2: Movies rated by users in decreasing order of ratings received by movies

We observe that the movie with the highest number of ratings received 329 ratings, while more than half of the movies received less than 4 ratings. There are also more 3000 movies that received only 1 rating. This phenomenon implied again the sparsity of the rating matrix and the difficulty of building a recommender system.

One thing to note here, however, is that this graph shows the **highest number of ratings** received by a movie, and not the movie that received the **highest ratings**. This suggest that the movie with small index in this histogram may not necessarily be the most well-received or best-rated movie, it is simply the movie that most people rated. However, from Question 2 we saw that very few movies had ratings between 0-2.5, so the chances of this movie being well-received are quite high.

QUESTION 4:

We extract unique users and then count the number of movies that each user rated. After that the users were grouped in a manner where the user with the first index had rated the highest number of movies, then the second one had the second highest number and so on and so forth. The graph is a monotonically decreasing function as shown in Fig. 3.

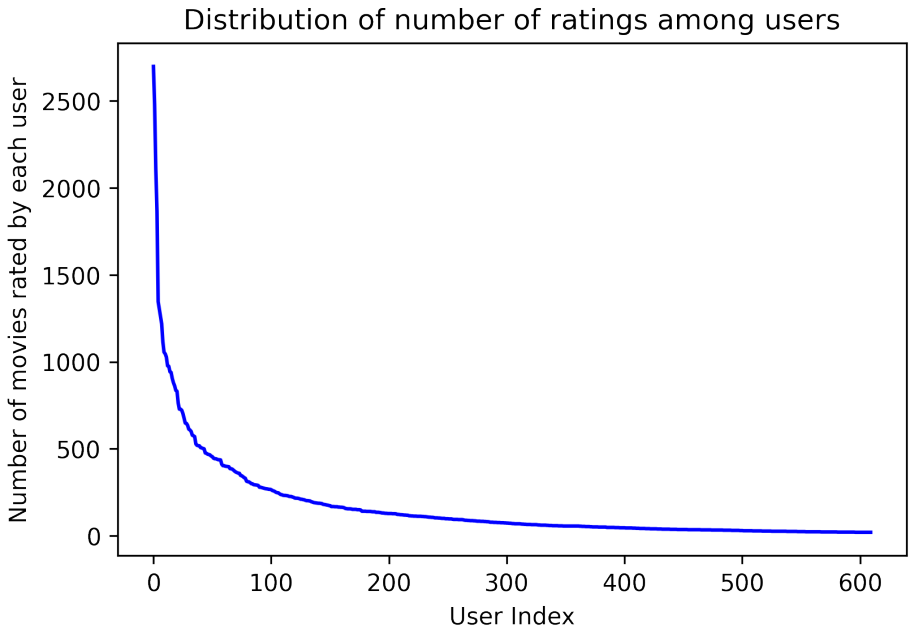

Figure 3: Movies rated by users in decreasing order of ratings given by users

The first 12 users rated over 1000 movies and there are more than half users who rated less than 100 movies, but even the users who rated least movies have rated 20 movies.

The users who watch few movies are the ones whom recommender systems should focus on, since these systems could analyze what kind of movies do they watch, what kind of movies do they favourably rate so that the system could predict and suggest movies to them so that these users could watch movies that they may potentially like.

QUESTION 5:

Fig. 2 from Question 3 shows the distribution of the number of ratings received by each movie and presented them in a decreasing order. As stated in Question 3, we observe the unbalanced distribution among all the movies, where few movies received a high amount of ratings while a majority of movies (more than half) received only a small amount of movies. The fast decreasing trend in the beginning and the long tail of the curve implied that item-based recommender system may not perform as well as user-based recommender system, because the inter-item correlation is

too sparse compared to the inter-user correlation (as shown in Fig. 3 and analyzed in Question 4).

Another observation of the curve that is easily mistaken is that, the movie that received highest number of ratings may or may not be the movies with top rating values, but considering there is a higher possibility for users to rate a movie with a rating of 3-5 according to Fig. 1, it is reasonable to predict that the movies with high number of ratings are popular among audience and thus also worth recommending to users who hadn't watched or rated them.

QUESTION 6:

We group movies with the same movieId in the raw dataset and calculated the variance of ratings of each movie. The histogram is plotted into binned histogram with bin width of 0.5, as shown in Fig. 4 and Fig. 5.

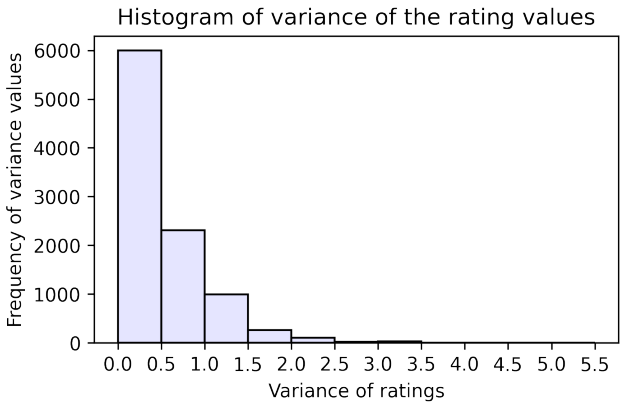

Figure 4: Histogram of variance of ratings

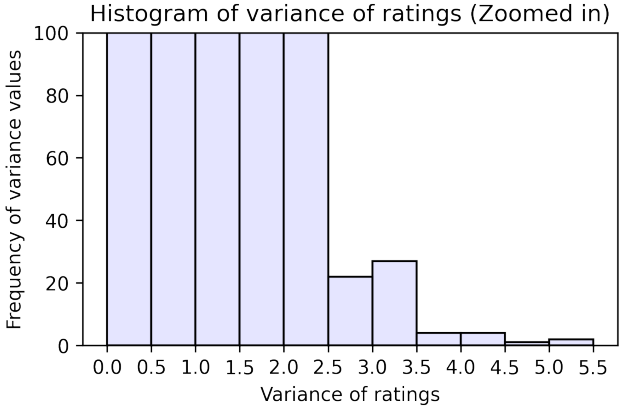

Figure 5: Zoomed-in histogram

Fig. 4 shows that most movies have a variance between 0 and 1 (more than 85%), signifying that most people tend to have the same review about a movie, irrespective of a genre. Very few movies have values of variance $\geqslant 2$ (less than 2%). These movies either have too few raters or the opinions towards which is significantly varied. The situation could occur when a movie tried some innovative way of story-telling, plotting, filming, etc, and therefore it may receive highly controversial opinions among audience.

4 Neighborhood-based collaborative filtering

4.2 Pearson-correlation coefficient

QUESTION 7:

Pearson-correlation coefficient captures the similarity between two rating vectors. The general formula is given by Equation 2:

$$Pearson(X,Y) = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}, \tag{2}$$

where

$\mu_X = \mathbb{E}[X]$ is the mean of set X,

$\mu_Y = \mathbb{E}[Y]$ is the mean of set Y,

$\sigma_X = \sqrt{\mathbb{E}[X^2] - (\mathbb{E}[X])^2}$ is the standard deviation of set X,

$\sigma_Y = \sqrt{\mathbb{E}[Y^2] - (\mathbb{E}[Y])^2}$ is the standard deviation of set Y.

The Pearson-correlation coefficient between users u and v is defined by Equation 3:

$$Pearson(u,v) = \frac{\sum\limits_{k \in I_u \cap I_v} (r_{uk} - \mu_u)(r_{vk} - \mu_v)}{\sqrt{\sum\limits_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \sqrt{\sum\limits_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \tag{3}$$

where the notations are as follows:

I_u: Set of item indices for which ratings have been specified by user u

I_v: Set of item indices for which ratings have been specified by user v

μ_u: Mean rating for user u computed using her specified ratings

r_{uk}: Rating of user u for item k

Therefore, we have

$$\mu_u = \frac{\sum\limits_{k \in I_u} r_{uk}}{|I_u|} \tag{4}$$

where $|I_u|$ is the number of values in I_u.

QUESTION 8:

$I_u \cap I_v$ represents the intersection set of indices which are common to both users u and v. In plain words, it is the set of movies for which both users u and v have rated.

$I_u \cap I_v = \varnothing$ is very likely to be true for random user u and v because the rating matrix R is very sparse. It's totally possible that there is probably no common movie for which both users u and v rated, hence their respective sets are mutually exclusive, leading to an empty intersection set.

4.4 Prediction function

QUESTION 9:

The predicted rating of user u for item j, denoted by $\hat{r}_{uj}$ is given by Equation 5:

$$\hat{r}_{uj} = \mu_u + \frac{\sum\limits_{v \in P_u} Pearson(u,v)(r_{vj} - \mu_v)}{\sum\limits_{v \in P_u} |Pearson(u,v)|}, \tag{5}$$

where P_u is the set of k-nearest neighbors of user u, which means the set of k users with the highest Pearson-correlation coefficient with user u, Pearson-correlation coefficient formula is given by Equation 3.

The reason why the Equation 5 includes mean-centering of the term $(r_{vj} - \mu_v)$ and $(\hat{r}_{uj} - \mu_u)$ is that, the general rating habits of each user may be different. Some users may tend to rate highly for all the movies they rated, some other users may rate all items poorly. Therefore, the mean-centering method is used to reduce the impact of different rating habits.

4.5 k-NN collaborative filter

k-NN or k-Nearest Neighbours refers to one of the classification methods in Machine Learning. It aims to classify a data point as one out of the given classes (usually 2 in number) by means of its neighbours.

All items for a class are initialized to 0 for each data point. An odd value of k is chosen (3, 5, 7 etc.) and so k number of nearest neighbours are considered. Then each neighbour's class is identified and the number for that particular class is incremented. Once this is done, the class with the highest number for this point is identified. This data point is then assigned the label of that class. In case two out of all the classes have the highest number, then either one out of these two classes is randomly assigned to the data point.

One thing to note about k-NN is that it tends to have a low train error but a high test error, and so in many cases it may not be the most accurate method for a recommender system.

4.5.1 Design and test via cross-validation

QUESTION 10: kNN on whole dataset

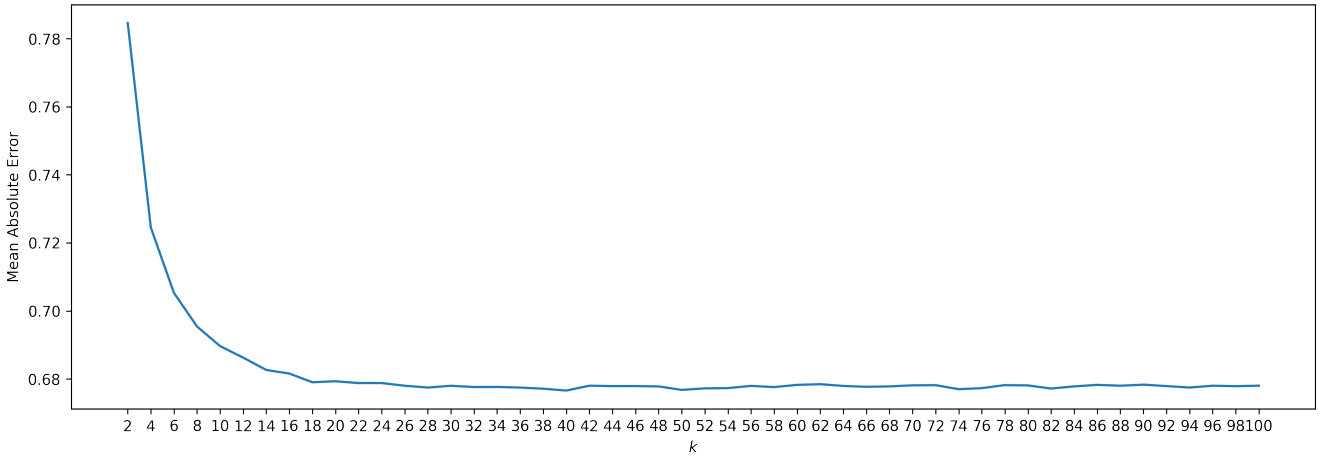

Figure 6: Mean Absolute Error of KNN vs k

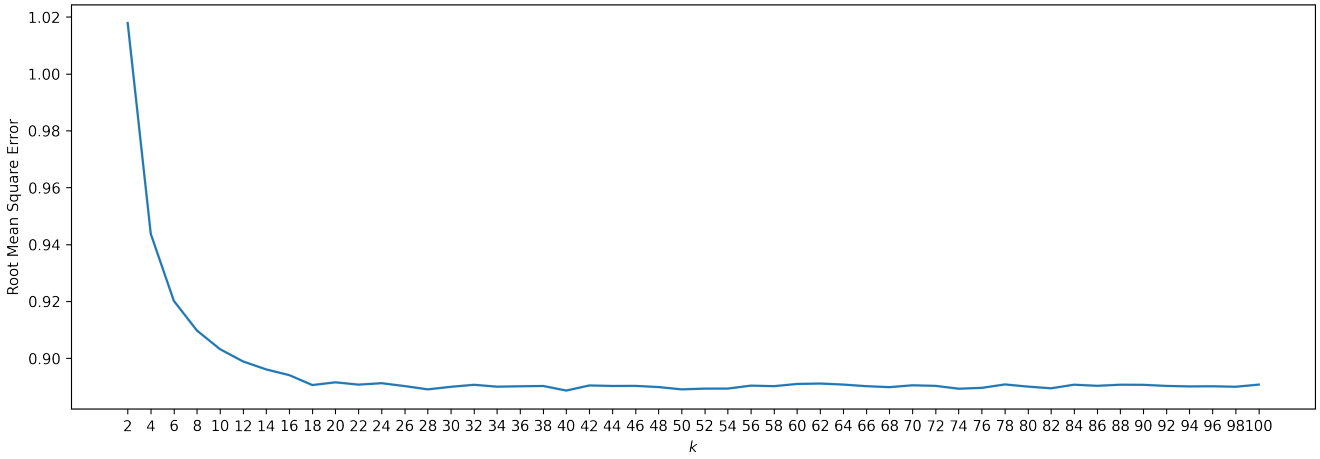

Figure 7: Root Mean Square Error of KNN vs k

In Figure 6 and Figure 7, we can see there is a downward trend in the Mean Absolute Error until k reaches about 20. This is mainly due to the fact that for small values of k, the kNN model lacks the capacity to accurately represent the data it was trained on. This leads to lower error until reaching a point where the model has sufficient capacity to represent the data. From this point on, increasing k yields no additional benefits.

QUESTION 11:

Steady state behavior for MAE and RMSE was achieved at $k = 20$. The steady state values for MAE and RMSE are: $MAE_{min} = 0.6793$, $RMSE_{min} = 0.8915$.

4.6 Filter performance on trimmed test set

QUESTION 12: kNN on popular movies

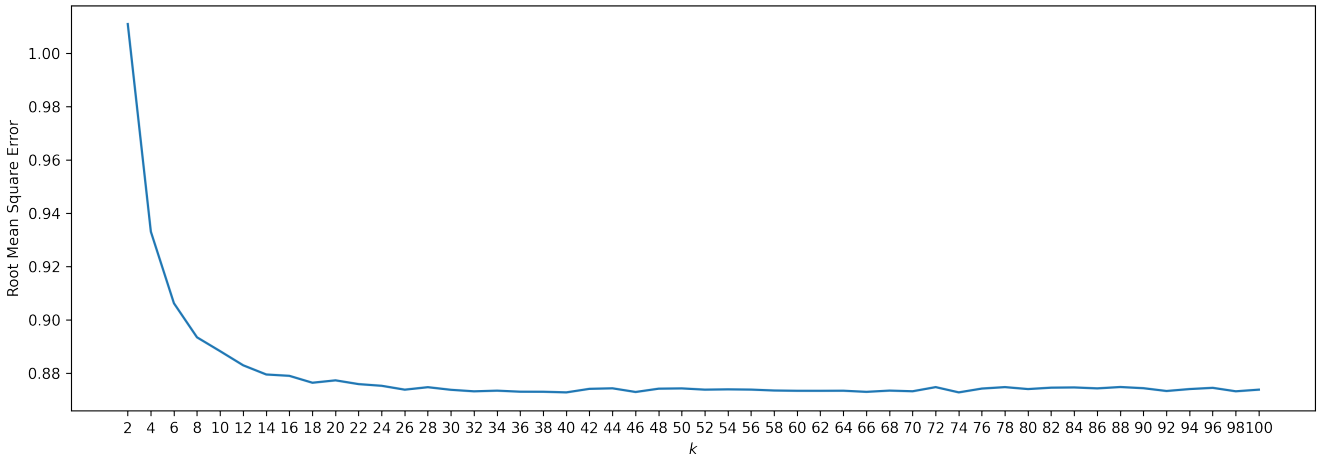

Figure 8: Root Mean Square Error of KNN on popular movies vs k

The minimum RMSE is 0.8727 at $k = 40$.

For the same reason given in Question 10, the RMSE shown in Figure 8 displays the same decreasing pattern until a steady state value is achieved at about $k = 20$.

QUESTION 13: kNN on unpopular movies

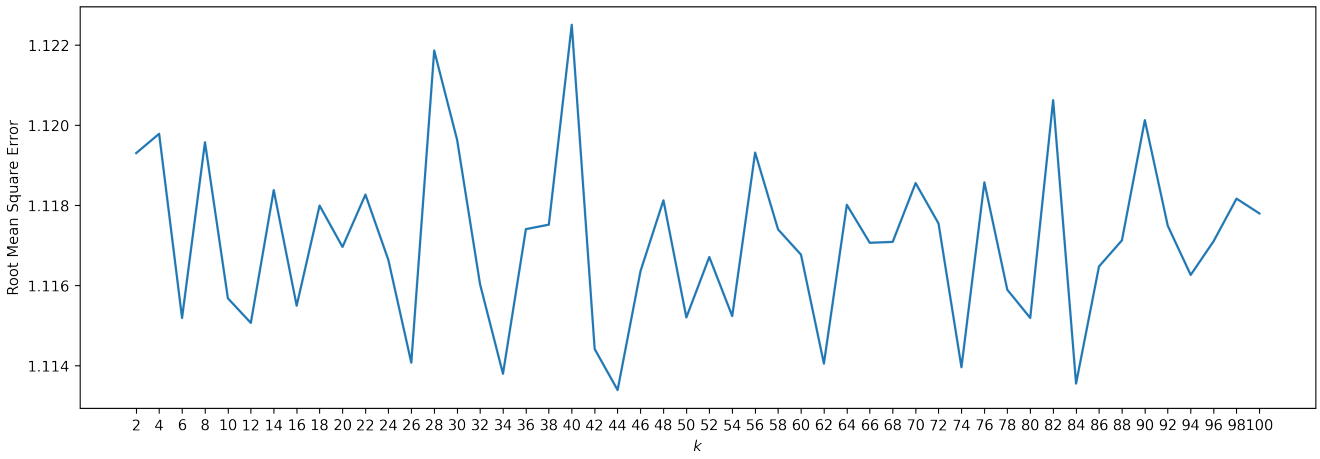

Figure 9: Root Mean Square Error of KNN on unpopular movies vs k

The minimum RMSE is 1.1134 at $k = 44$.

Compared to figure in Question 10 and 12, the RMSE shown in Figure 9 is much noisier due to the small size of the trimmed test set. The recommender system in problem is trained on the full training dataset where popular films have more ratings and unpopular films have less known ratings, therefore, during the test, the recommender system mostly predict user's favour on those unpopular films based on their rating on popular films, which explained the reason why the testing performance is poor no matter how much the value of k is.

QUESTION 14: kNN on high-variance movies

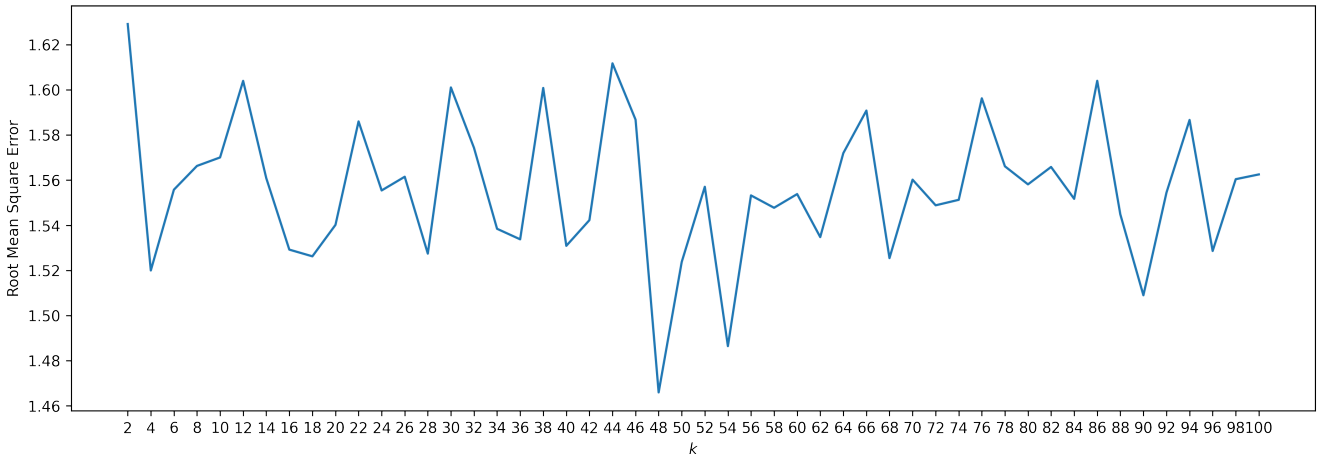

Figure 10: Root Mean Square Error of KNN on high-variance movies vs k

The minimum RMSE is 1.4660 at $k = 48$.

For the same reason given in Question 13, the RMSE shown in Figure 10 is very noisy, because in the sparse rating matrix, the given ratings for high-variance movies are very few, therefore, making recommendations of high-variance movies when the majority of training is based on low-variance movies is very likely to give a poor performance whatever the value of k is.

4.6.1 Performance evaluation using ROC curve

QUESTION 15: ROC curve for kNN

From Fig. 7 in Question 10 and 11, we find the best k to use for k-NN is $k = 20$, which is used to plot the ROC curves for threshold values [2.5, 3, 3.5, 4] in Fig. 11.

We notice that the four curves are almost identical, with slight difference in the AUC value: the higher threshold is, the lower AUC value is.

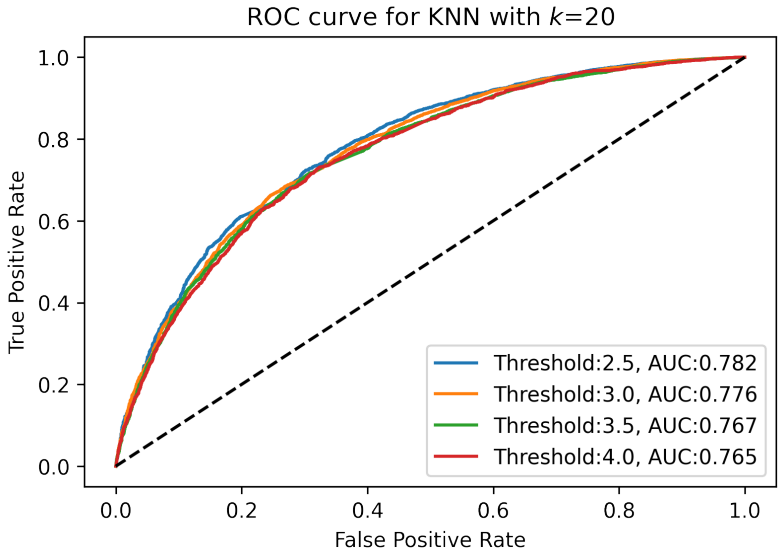

Figure 11: ROC Curve for KNN with $k = 20$ and various thresholds

5 Model-based collaborative filtering

5.2 Non-negative matrix factorization (NNMF)

Non-Negative Matrix Factorization (NNMF) is a state-of-art algorithm that is used to extract features[1]. A lot of features are somewhat ambiguous and so it becomes difficult to infer meaningful information from them. NNMF combines these features to help produce meaning and features that could be incorporated for classification/prediction.

It has been used widely for dimensionality reduction and feature extraction of non-negative data[2]. The main difference between NMF and other factorization methods, such as SVD, is the non-negativity, which allows only additive combinations of intrinsic 'parts', i.e. the hidden features. The positive additive combination is naturally more intuitive than the negative one.

QUESTION 16:

The optimization formulation in NNMF without regularization terms is given by Equation 6:

$$\min_{U,V} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2, \tag{6}$$

where $U \geqslant 0$, $V \geqslant 0$, $W_{ij} = 1$ if and only if the term r_{ij} is known otherwise $W_{ij} = 0$.

The convexity of Equation 6 is difficult to judge due to the existence of two matrices U, V.

So in order to analyse the optimization problem in an easier way, we first keep the value of U fixed. Essentially, we'll assign a fixed U_0 to U so that it becomes a single variable optimization problem. Equation 6 thus becomes:

$$\min_V \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(r_{ij} - (U_0 V^T)_{ij})^2, \tag{7}$$

where U_0 is a constant positive matrix.

Then we can formulate this as a least squares problem as below:

$$\min_V \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(r_{ij} - U_{0,i} V_j)^2, \tag{8}$$

where $U_{0,i}$ is the i^{th} row of the matrix U_0, V_j is the j^{th} column of the matrix V.

Therefore, for each j, the problem is to minimize the term $(r_{ij} - U_{0,i} V_j)^2$, which is basically a least squares problem for V_j.

Using iterative algorithm, we can calculate the optimal matrix $\cap V$ with fixed U_0, and then plug back into Equation 6, solve for U with fixed $\cap V$, which is again a convex problem for the only variable matrix U.

The real optimal combined solution for both U and V can be approached after certain amount of iterations of optimal U and V.

5.2.1 Prediction function

The prediction rating of user i for item j using NNMF is given by Equation 9:

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js}, \tag{9}$$

where k is the number of latent factors in NNMF.

5.2.2 Design and test via cross-validation

QUESTION 17: NNMF on whole dataset

We designed a NNMF-based collaborative filter to train and test on the entire dataset. The number of latent factors k is swept from 2 to 50 in step sizes of 2, as shown in Fig. 12 and Fig. 13.

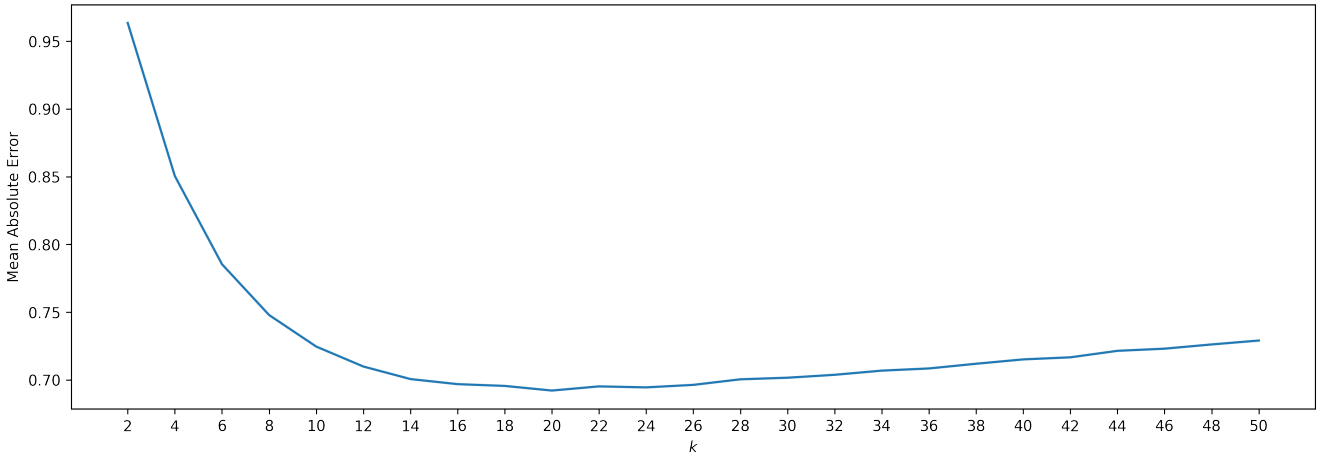

Figure 12: Mean Absolute Error of NNMF vs k

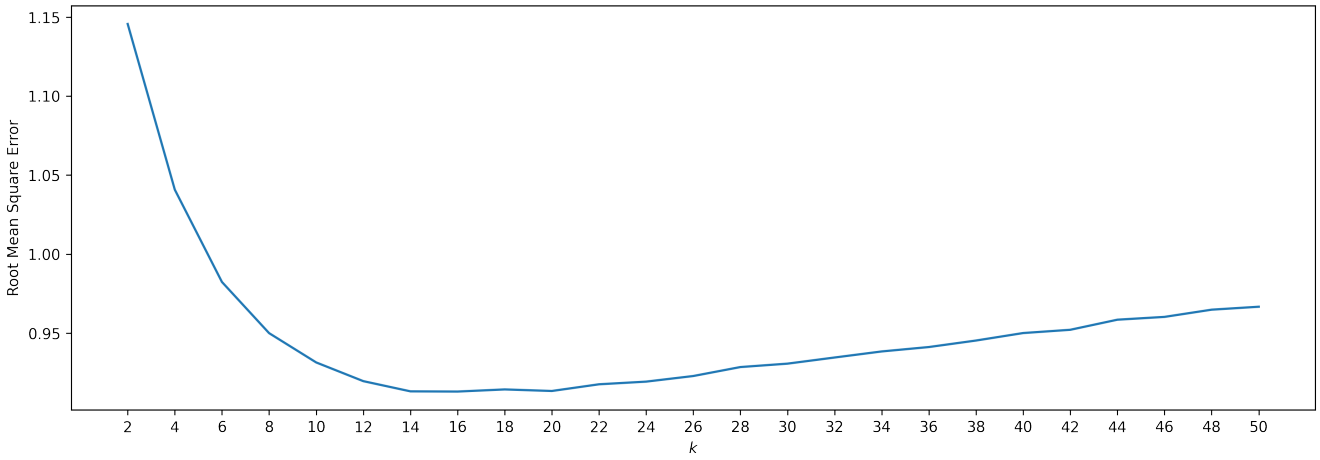

Figure 13: Root Mean Square Error of NNMF vs k

As shown in Fig. 12 and Fig. 13, there is a monotonically decreasing function observed, which proves that as we increase the number of latent factors k, the errors decrease. There is a slight increase after $k = 20$.

Furthermore, once can also observe that while RMSE and MAE have different graphs, the general trend in them is same but the graphs have some differences in the extent of error, which could naturally be attributed to the fact that they have different methodologies of calculating the error.

QUESTION 18:

According to results from Question 17, we find that:

The minimum MAE is 0.6922 at $k = 20$

The minimum RMSE is 0.9131 at $k = 16$

The number of optimal factors is very close to the number of movie genres (18), though not exactly the same, but already close enough to show that the number of latent factors k in NNMF provides interpretability during the dimensionality reduction.

5.2.3 NNMF filter performance on trimmed test set

QUESTION 19: NNMF on popular movies

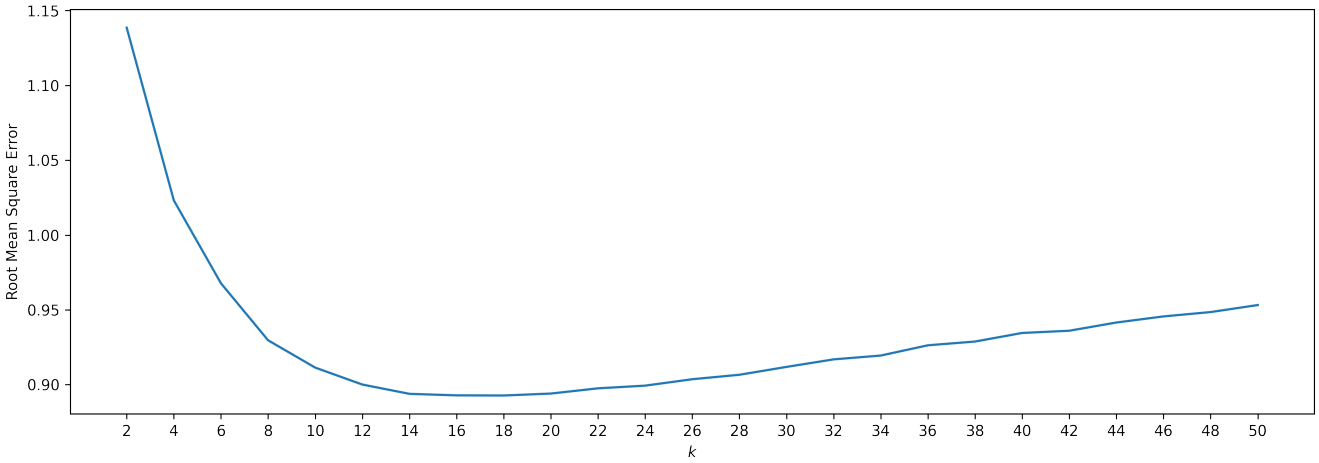

Figure 14: Root Mean Square Error of NNMF on popular movies vs k

The minimum RMSE is 0.8928 at $k = 18$.

The general trend of the RMSE curve in Fig. 14 is very similar to that of Fig. 13, with a decreasing trend in the first and a slight increasing tendency when k is larger than certain value. The minimum value of RMSE is achieved at $k = 18$, which is also the same as the number of movie genres, proving the interpretability of the NNMF filter.

The curve makes sense because if a movie is popular, it's likely to be liked by a majority of the users and thus arriving at an accurate prediction is easier as compared to unpopular and high-variance dataset, which explained the reason why the curve looks similar to figures in Question 17.

QUESTION 20: NNMF on unpopular movies

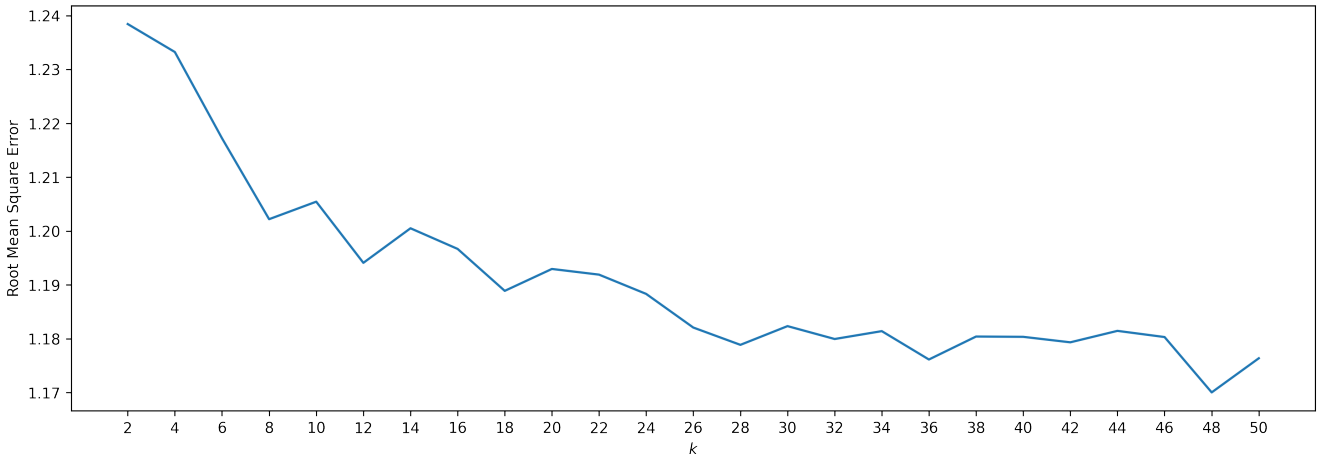

Figure 15: Root Mean Square Error of NNMF on unpopular movies vs k

The minimum RMSE is 1.1700 at $k = 48$.

As can be seen from Fig. 15, there is an almost monotonically decreasing function observed, which proves that as the number of latent factors k increases, the error of the recommender system decreases, indicating that the performance becomes better.

This does make sense as the ratings received by unpopular movies are too few ($\leqslant 2$ ratings) so it's very difficult to make an accurate estimation of prediction for each user. Therefore, the larger the number of latent factors k is, the more information is drawn from the sparse rating matrix, therefore, it is more likely to make better prediction for users.

QUESTION 21: NNMF on high-variance movies

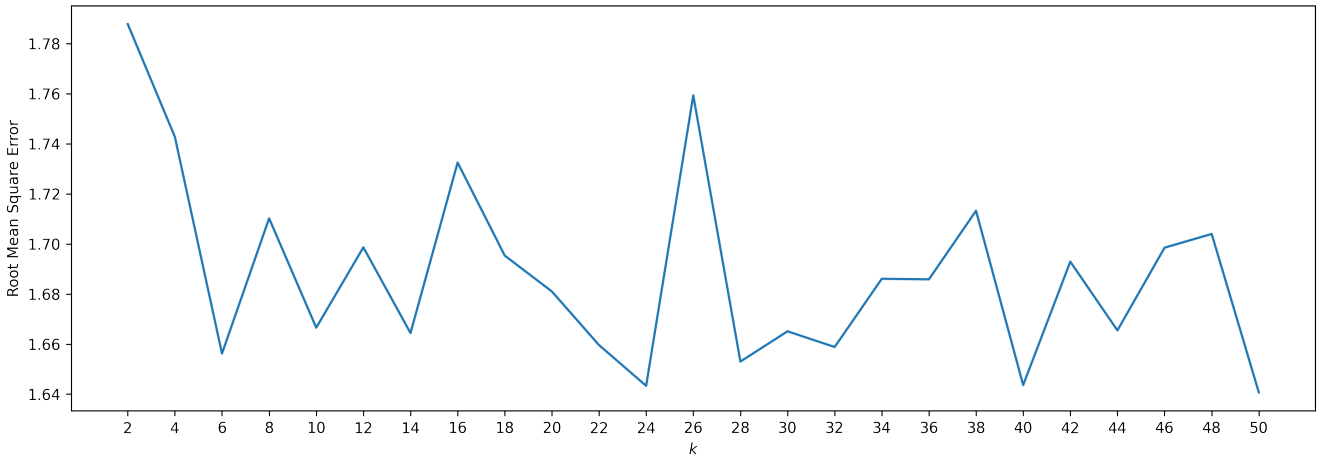

Figure 16: Root Mean Square Error of NNMF on high-variance movies vs k

The minimum RMSE is 1.6407 at $k = 50$.

As we can observe from Fig. 16, the RMSE curve is very noisy among different k, the reason is similar to the explanation in Question 14, which is due to the low number of ratings and inconsistent opinions towards these high-variance movies.

The inaccurate prediction result is reasonable since if a movie has high variance in the ratings, it becomes more difficult to observe the general trend for that (as some may like it and some won't and the extent of like/dislike varies greatly) so predicting becomes a tough job, even more difficult and unstable than the prediction for unpopular movies. Therefore, it also makes sense that the performance would be slightly better when k is larger (in our case, k reaches the maximum value in the sweeping range) and more information is drawn for training and testing.

5.2.4 Performance evaluation using ROC curve

QUESTION 22: ROC curve for NNMF

From Fig. 12 and Fig. 13 in Question 17 and analysis in Question 18, we decide that the optimal k for NNMF is the k that is able to provide the interpretability and represent the number of movie genres. Therefore, since we get minimum MAE at $k = 20$ and minimum RMSE at $k = 16$, and considering the number of movie genres is 18, we decide to use $k = 18$ as our optimal value of k for NNMF and plot the ROC curve accordingly, as shown in Fig. 17.

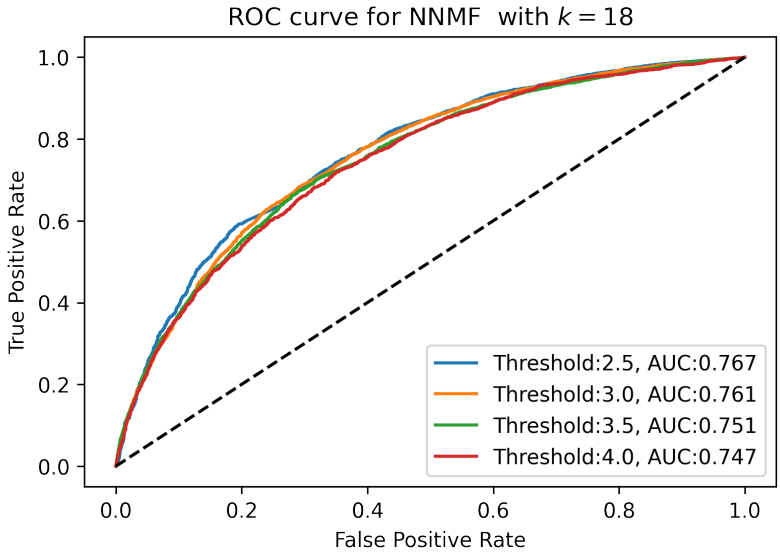

Figure 17: ROC Curve for NNMF with $k = 18$ and various thresholds

Similar to Fig. 11 in Question 15, the ROC curves and their corresponding AUCs in Fig. 17 show

a decreasing trend where larger value of threshold leads to smaller value of AUC, which makes sense. As a model learns the dataset well, its predictions become more accurate and the error would decrease (represented by the area below True Positive Rate - False Positive Rate curve).

5.2.5 Interpretability of NNMF

QUESTION 23:

For this question, Ratings matrix R was calculated as having 610 rows (for 610 users) and 9724 columns (for 9724 movies) so each item R_{ij} correspond to a rating given by user i to movie j (the value is set to 0 if the corresponding rating is missing). NMF was applied on R to get non-negative matrices U and V with k latent factors. We rearrange the rows of the matrix V in descending orders for each column, and report the top 10 movies for the first three columns of the matrix V to analyse the interpretability of NNMF.

Column Index = 0:

The genres of the top 10 movies were:

Comedy|Horror|Sci-Fi|Thriller
Action|Adventure|Horror|Sci-Fi
Action|Sci-Fi
Action|Adventure|Sci-Fi|Thriller
Action|Adventure|Drama|Sci-Fi|Thriller
Action|Drama|Horror
Action|Comedy|Crime|Mystery
Action|Drama|Sci-Fi
Drama|Fantasy|Horror|Romance
Action|Horror|Sci-Fi

{'Action': 8, 'Adventure': 3, 'Comedy': 2, 'Crime': 1, 'Drama': 4, 'Fantasy': 1, 'Horror': 5, 'Mystery': 1, 'Romance': 1, 'Sci-Fi': 7, 'Thriller': 3}

We notice that the genre "Action" occurs 8 times out of 10 movies, "Sci-fi" 7 times, "Horror" 5 times, with a few combination from other genres. It is confident to predict that this column of the matrix V represent most of the movies that are most outstanding in the combined genre of Action and Sci-fi.

Column Index = 1:

The genres of the top 10 movies were:

Crime|Drama
Comedy|Drama

Comedy|Drama
Comedy|Drama|Romance
Drama
Comedy|Drama
Comedy|Drama|Romance
Adventure|Animation|Comedy
Drama
Comedy|Crime

{'Adventure': 1, 'Animation': 1, 'Comedy': 7, 'Crime': 2, 'Drama': 8, 'Romance': 2}

In this example column, we notice the genre "Drama" occurs 8 times, "Comedy" 7 times, with a few other combinations. We can predict that this column of V most likely represent the common features for comedy-drama movies.

Column Index = 2:

The genres of the top 10 movies were:

Action|Adventure|Drama|Fantasy
Adventure|Fantasy
Adventure|Fantasy
Action|Adventure|Sci-Fi
Adventure|Animation|Children|Romance|Sci-Fi
Action|Adventure|Sci-Fi
Action|Adventure|Sci-Fi|IMAX
Adventure|Animation|Children|Drama
Action|Crime|Drama|IMAX
Action|Sci-Fi|Thriller

{'Action': 6, 'Adventure': 8, 'Animation': 2, 'Children': 2, 'Crime': 1, 'Drama': 3, 'Fantasy': 3, 'IMAX': 2, 'Romance': 1, 'Sci-Fi': 5, 'Thriller': 1}

In this column example, we notice the genre "Adventure" occurs 8 times, "Action" 6 times, "Sci-fi" 5 times, with a few combinations from other genres. We can interpret the meaning of this column of matrix V represent common features among many Adventure-Action movies.

Most interestingly, after printing out the title of these top movies, we find out that the top three movies actually correspond to "The Lord of the Rings: The Return of the Kings", "The Lord of the Rings: The Fellowship of the Ring", and "The Lord of the Rings: The Two Towers". The fact that these movies belong to a same series further proves that the NNMF method is able to draw common features from a small collection of genre and offer more interpretability compared to models from other methods.

To conclude, according to the three examples that we analysed above, we confirm that the top movies from the same column of the matrix V tend to belong to a small collection of genres.

The number of latent factors k influences the number of features drawn from the raw rating matrix. In our case, the "feature" here means a certain combination of the genres, such as action-sci-fi movies, comedy-drama movies, adventure-action movies, etc. The larger the value of k is, the more specific the features are, but it is not necessary true that the large k is the better the model performance is.

5.3 Matrix factorization with bias (MF with bias)

Matrix decomposition refers to methods that reduce a matrix into constituent parts[3]. This is an effective way to simplify the calculations for complicated matrix operations.

A simple analogy is like factorizing 16 as 2×8. For this reason, Matrix decomposition methods are also known as Matrix Factorization methods. These form the base for linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

Since we already worked on matrix factorization with NNMF, in this part, we are going to work on MF with bias, which means the optimization formulation of Matrix Factorization would include the bias terms of the users.

5.3.2 Design and test via cross-validation

QUESTION 24: MF with bias on whole dataset

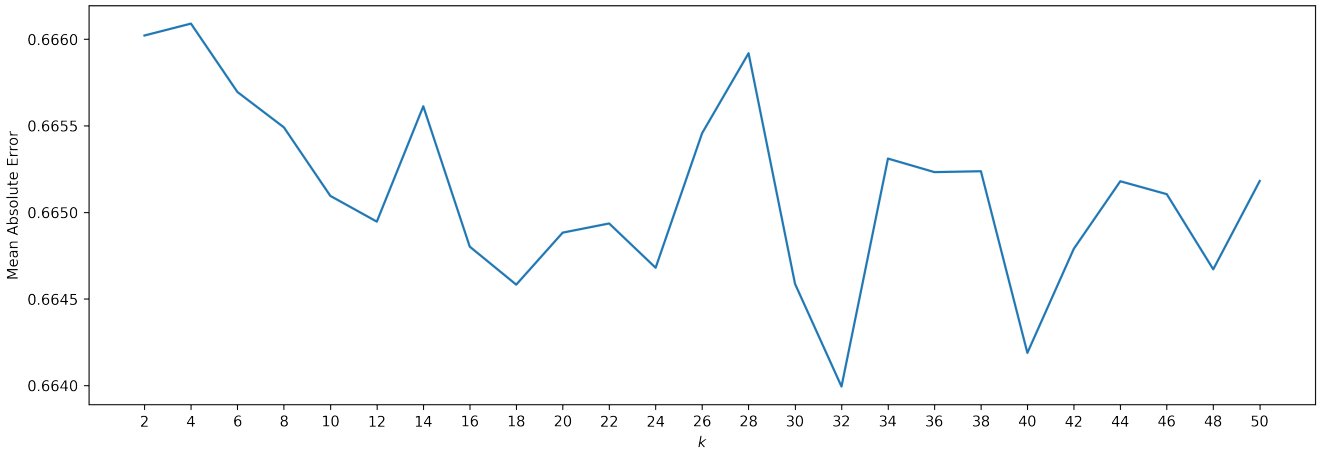

Figure 18: Mean Absolute Error of SVD vs k

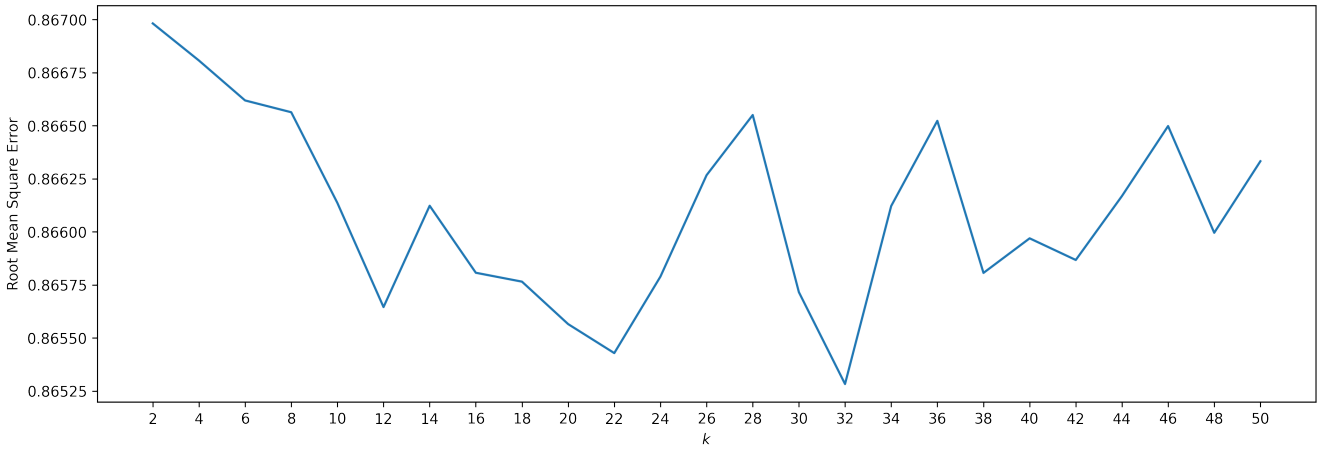

Figure 19: Root Mean Square Error of SVD vs k

We implemented the MF with bias collaborative filter using the function "SVD" from the library "surprise". The number of latent factors k is swept from 2 to 50 in step sizes of 2, as shown in Fig. 18 and Fig. 19.

From the two figures, we can see that both MAE and RMSE curves are very noisy, but the overall trend is decreasing as k increases. The minimum value of MAE and RMSE are both reached when $k = 32$.

QUESTION 25:

According to results from Question 24, we find that:

The minimum MAE is 0.6640 at $k = 32$

The minimum RMSE is 0.8653 at $k = 32$.

5.3.3 MF with bias filter performance on trimmed test set

QUESTION 26: MF with bias on popular movies

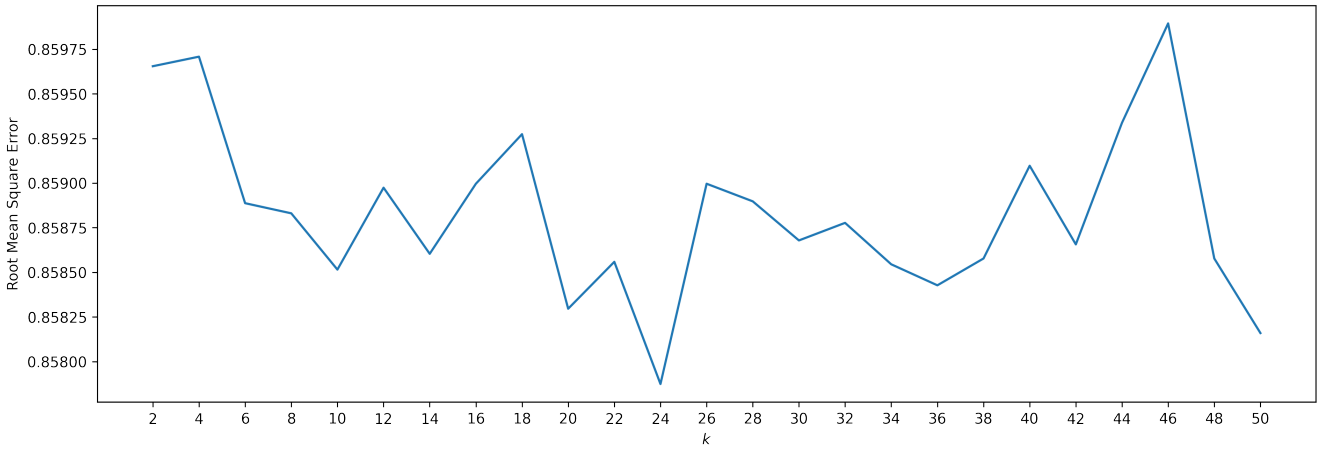

Figure 20: Root Mean Square Error of SVD on popular movies vs k

The minimum RMSE is 0.8579 at $k = 24$.

For the same reason given in Question 24, the RMSE shown in Figure 20 displays a similar noisy pattern, though the minimum RMSE value is slightly smaller.

QUESTION 27: MF with bias on unpopular movies

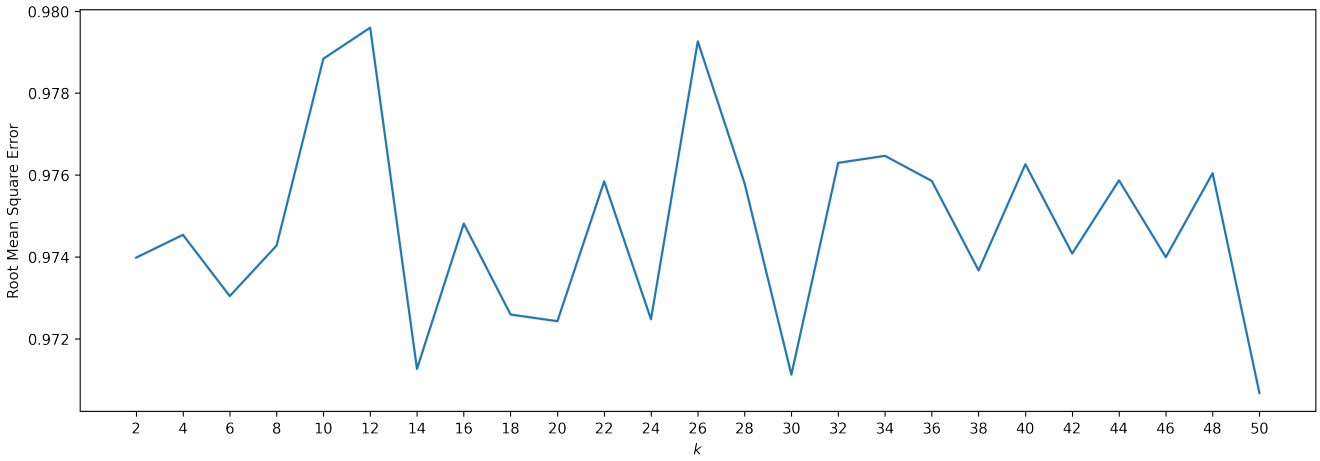

Figure 21: Root Mean Square Error of SVD on unpopular movies vs k

The minimum RMSE is 0.9707 at $k = 50$.

The RMSE curve in Fig. 21 displays a similarly noisy pattern compare to Question 24 and 26.

QUESTION 28: MF with bias on high-variance movies

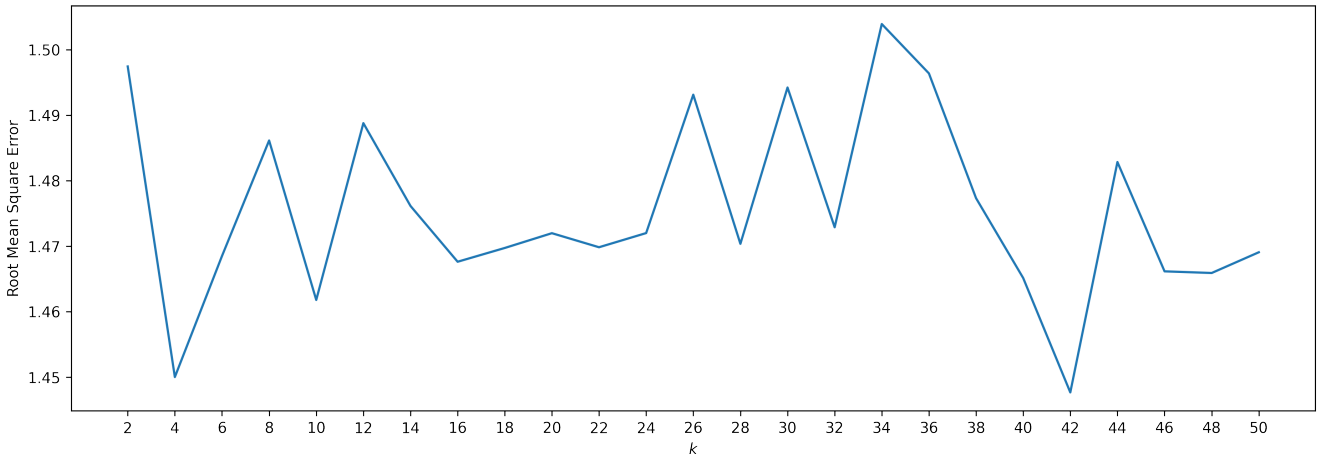

Figure 22: Root Mean Square Error of SVD on high-variance movies vs k

The minimum RMSE is 1.4477 at $k = 42$.

The RMSE curve in Fig. 22 displays a noisy pattern, which is similar to the situation with popular movies and unpopular movies. This shows that the model is consistent to number of ratings that a movie received. Therefore, this method should be most stable when it is applied to other highly sparse rating matrix.

5.3.4 Performance evaluation using ROC curve

QUESTION 29: ROC curve for MF with bias

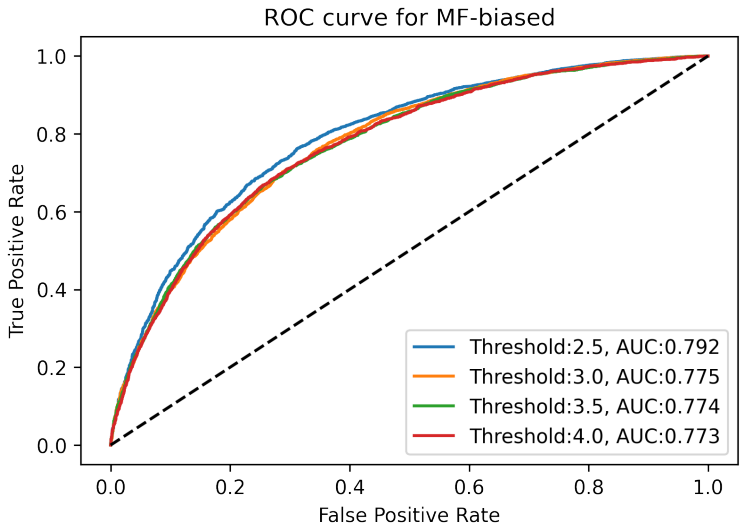

Figure 23: ROC Curve for MF with bias with $k = 32$ and various thresholds

From Fig. 18 and Fig. 19 in Question 24 and analysis in Question 25, we find the optimal k for MF with bias is $k = 32$. Therefore, we used $k = 32$ to plot ROC curve, as shown in Fig. 23.

Similar to Fig. 11 in Question 15 and Fig. 17 in Question 22, the AUC below the ROC curve slightly decreases as the threshold value increases though the difference is very trivial.

6 Naive collaborative filtering

6.2 Design and test via cross-validation

Naive Collaborative Filter uses the naive way of estimating the ratings for a given movie. The prediction function is given by Equation 10:

$$\hat{r}_{ij} = \mu_i \tag{10}$$

where $\hat{r}_{ij}$ is the estimated rating for movie j given by user i and μ_i is the mean of all the ratings given by user i.

In implementation, there is no training process since the prediction of all the testing datas are based on the average rating of the training data, grouped by the same userId. To evaluate the performance of the naive filter, we calculated the average RMSE between the predictions and the ground truth labels among all the ten-folds via cross-validation.

QUESTION 30: Naive filter on whole dataset

RMSE = 0.9342

The value of RMSE is calculated on the whole original rating dataset. The value came out to be close to but larger than what we got for sophisticated models like kNN, SVD, NNMF and MF with bias, which makes sense. It's the mean of the user's rating. On average, a user rates movies between 0 to 5 so mean would be 2.5. To say that the user would always think of the movie as an average one is an erroneous prediction. Hence Naive filter reneders a larger error.

6.3 Naive collaborative filter performance on trimmed test set

QUESTION 31: Naive filter on popular movies

RMSE = 0.9646

The value of RMSE is calculated on the popular movie dataset. The value is close to but larger than what we got for sophisticated models like kNN, SVD, NNMF and MF with bias, and also larger than the result from Question 30. These all make sense because when estimating test movies

using only the mean value of all ratings that a user had rated, the prediction would not be very precise, especially on trimmed test set.

QUESTION 32: Naive filter on unpopular movies

RMSE = 0.9689

The value of RMSE is calculated on the unpopular movie dataset. The value is close to what we got for sophisticated models like kNN, SVD, NNMF and MF with bias. This result is expected with the same reason as in Question 31. Though this is lower than kNN, SVD etc. because for unpopular movies, the ratings as well as mean rating would lie in the lower range and so the prediction as mean wouldn't be that far off from the actual value

QUESTION 33: Naive filter on high-variance movies

RMSE = 1.1383

The value of RMSE is calculated on the high-variance movie dataset. The value is close to what we got for sophisticated models like kNN, SVD, NNMF and MF with bias. However, the value is the lowest compared to other methods, maybe because when there is a high variance in the ratings for a movie, the average would be something in between the two extremes, thus counteract the influence of either extremes. As Naive method involves mean of the user, that is an average term, this aligns with the filter behaviour and hence probably has the least error. But it has the greatest error compared to popular and unpopular because of high variation and hence there's greater difficulty in prediction.

Table 1: Minimum results from all methods

Filter	MAE (whole)	RMSE (whole)	RMSE (pop)	RMSE (unpop)	RMSE (high-var)
kNN	0.6793	0.8915	0.8727	1.1134	1.4659
NNMF	0.6922	0.9131	0.8928	1.1700	1.6407
MF w/ bias	**0.6640**	**0.8653**	**0.8579**	0.9707	1.4477
Naive	–	0.9342	0.9646	**0.9689**	**1.1383**

For ease of comparing the results from all the methods, we wrote all the minimum results in the same table, as shown in Table 1. All the smallest values in each column is written in bold font.

It is quite clear to see that, when testing on the whole dataset and popular movies, MF with bias gives the best performance, while on unpopular movies and high-variance movies, the naive filter returns the smallest RMSE value.

It is also reasonable to see that, for all the methods, the RMSE on whole dataset are quite similar

to the RMSE on popular movies, while unpopular movies return larger RMSE, and high-variance movies even larger.

The reason why high-variance movies get the largest error is that the number of popular movies (4980) and the number of unpopular movies (4744) both take about half of the whole dataset, while there are only 40 high-variance movies. The small sample of test dataset is not well targeted or represented during the training process, therefore, the prediction is quite inaccurate.

As for the unpopular movies, since these movies only received less than 2 ratings each, it is quite difficult for the model to estimate the quality and feature of the movie, leading to high error in the prediction results.

Since ratings received by popular movies take up most of the rating matrix, it makes sense that the prediction results on popular movies are very close to that on the whole dataset.

7 Performance comparison

QUESTION 34:

In order to compare the performance of the various collaborative filters that we designed in the earlier sections, we plot the ROC curves for kNN, NNMF, and MF with bias based collaborative filters in Fig. 24. The optimal k in each case is as following: $k = 20$ for kNN (from Question 11), $k = 18$ for NNMF (from Question 18), and $k = 32$ for MF with bias (from Question 25).

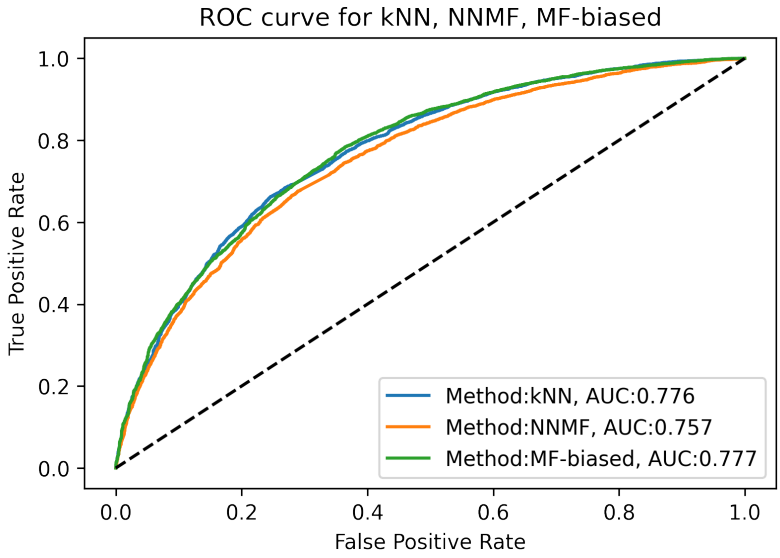

Figure 24: ROC Curves for kNN, NNMF, MF with bias

We can see that MF with bias returns the highest score of AUC, showing that it has the best performance in predicting the ratings of the movies, which is also confirmed by Table 1.

We also observe that the performance of kNN collaborative filter is very close to that of MF with bias as their ROC curves almost always overlap and their AUC scores are the same until the third decimal. The performance of NNMF is clearly worse than the other two methods, which can be seen from the fact that the curve is lower almost everywhere and the AUC value is the smallest.

8 Ranking

8.2 Evaluating ranking using precision-recall curve

Precision and recall are defined by Equation 11 and Equation 12:

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|}, \tag{11}$$

$$Recall(t) = \frac{|S(t) \cap G|}{|G|}, \tag{12}$$

where $S(t)$ is the set of movies of size t recommended to the user, G is the ground-truth set of movies liked by the user.

QUESTION 35:

According to Equation 11, Precision measures the percentage of how many items are correctly predicted among all the prediction.

According to Equation 12, Recall measures the percentage of how many items are correctly predicted among all the real-liked items.

As a rough analogy, precision can be considered as homogeneity in a clustering problem, while recall can be considered as completeness in clustering problem.

In real-life application, precision is more important for recommender system because the recommendation size is limited, therefore, the model should be able to recommend as many as possible items that users will be interested and liked. In contrast, for more critical circumstances such as medical-related applications, recall score is more important because the size of prediction set can be large while the recommended items should be as accurate as possible to fall into the set where the user will need.

QUESTION 36: kNN ($k = 20$)

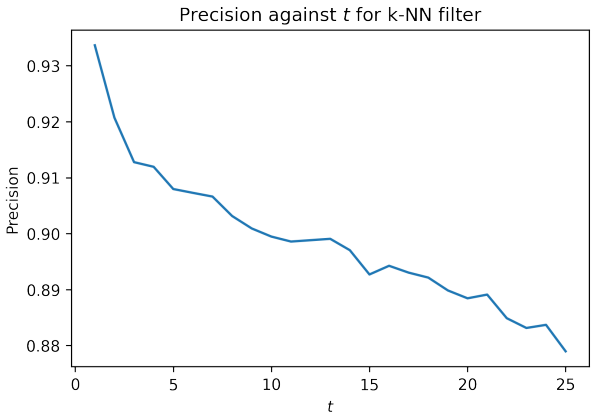

Figure 25: Precision-t for kNN

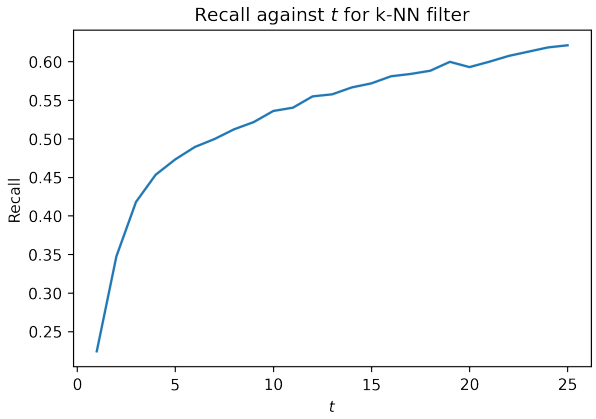

Figure 26: Recall-t for kNN

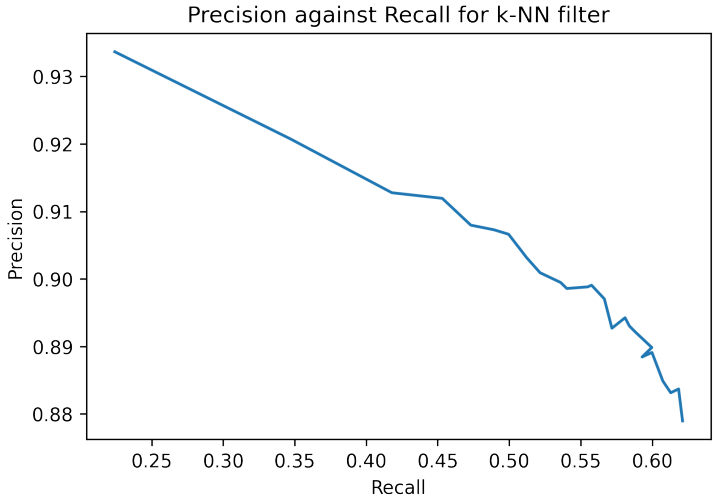

Figure 27: Precision-Recall curve for kNN

From Fig. 25 and Fig. 26, we can see that as t increases, precision score continuously decreases, and the recall score increases. Both trends change fast when t is below 5, and then change in a slower rate when t is larger than 5.

From Fig. 27, we can see as recall score increases, the precision score decreases steadily, which align with the shape of Fig. 25 and Fig. 26. The trend is close to linear relationship.

The shape of these curves are reasonable as when the number of recommendation grows larger, it is more likely to recommend a movie that the user would like, as well as a movie they may not like. Therefore, the precision deceases and recall increases, it is important to find an optimal value of t to balance the two scores.

QUESTION 37: NNMF ($k = 18$)

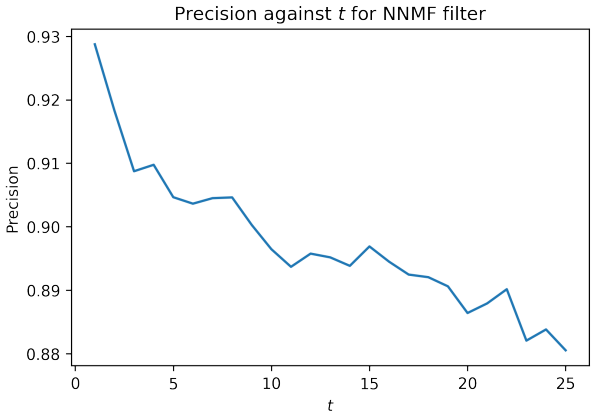

Figure 28: Precision-t for NNMF

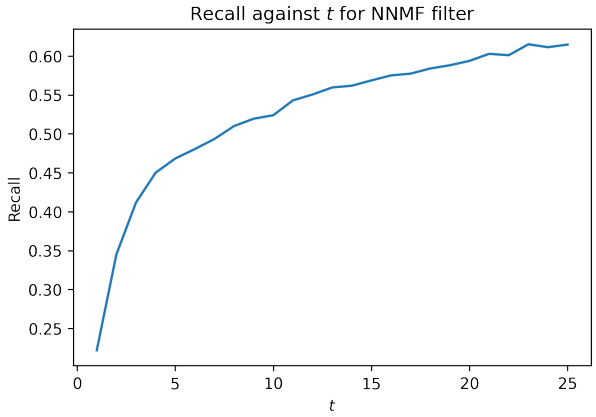

Figure 29: Recall-t for NNMF

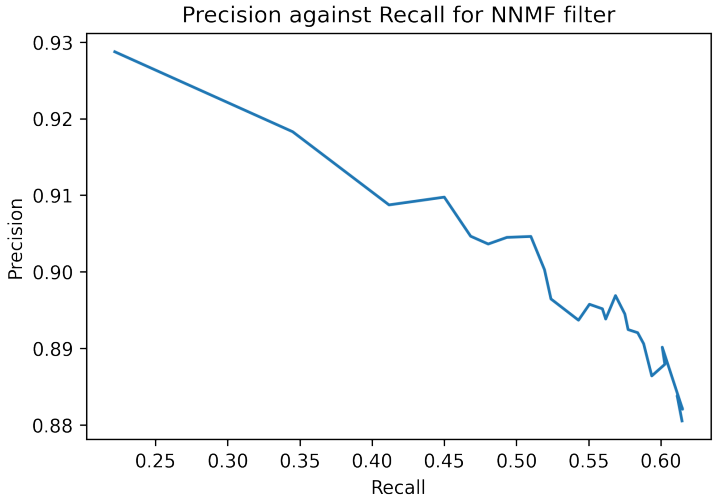

Figure 30: Precision-Recall curve for NNMF

The shape of the three curves for NNMF is very similar to that of kNN in Question 36, except that the precision-t curve in Fig. 28 looks a little bit more noisy. The respective explanation of the general trend of the three curves are also similar to that of Question 36.

QUESTION 38: MF with bias ($k = 32$)

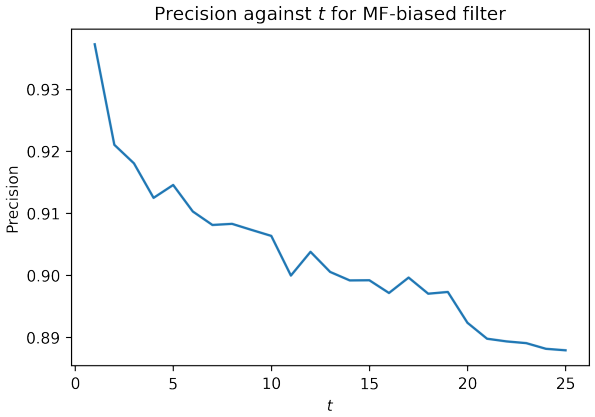

Figure 31: Precision-t for MF with bias

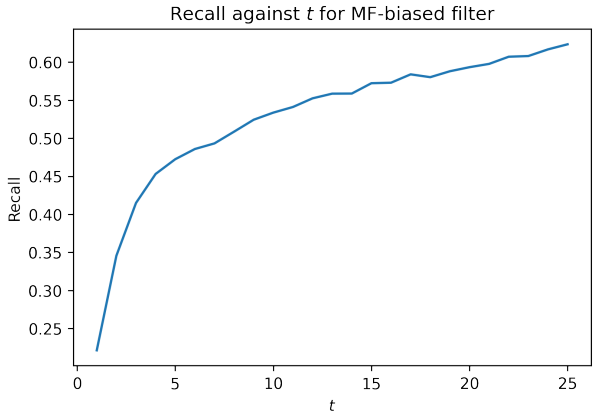

Figure 32: Recall-t for MF with bias

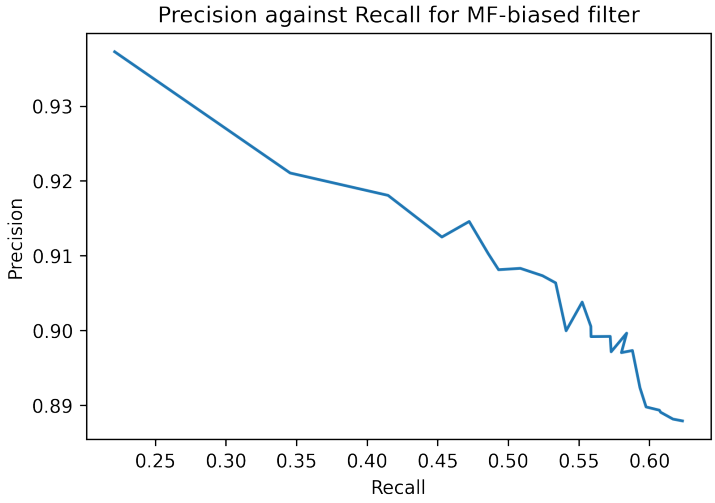

Figure 33: Precision-Recall curve for MF with bias

Similar to kNN and NNMF, for MF with bias, as t increases, the precision decreases and the recall increases, all the changing rate are also fast in the beginning stage and slower when t grows larger. The explanation is the same as in Question 36.

QUESTION 39: Precision-Recall Curve

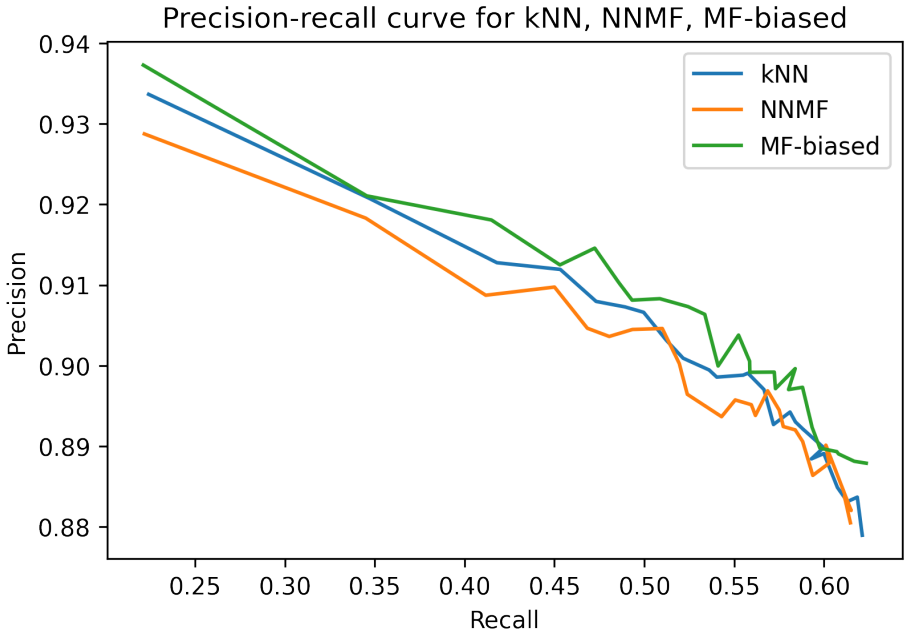

Figure 34: Precision-Recall curve for kNN, NNMF, and MF with bias

The performance of a prediction model can be estimated from precision-recall curve in that, for certain recall score, the higher the precision score is, the better the performance of the method is.

From Fig. 34, we can see that for certain recall that is smaller than 0.5, the performance of MF with bias is almost always best while NNMF the worst. When recall grows larger, which also means the value of t grows larger, the performance of all three models become more unstable, and the precision-recall curves would become very noisy.

Therefore, according to the ranking method and precision-recall curve in Fig. 34, the best recommender system is MF with bias, and the value of t should be limited to about 10 such that recall would remain below 0.5 in exchange for a high precision score. This conclusion is also consistent with the conclusion from RMSE analysis in Table 1 and the performance evaluation in Fig. 24 from Question 34.

References

1. NNMF: Non-Negative Matrix Factorization
2. BMCBioinformatics - NNMF
3. MF: Matrix Factorization