Large-Scale Data Mining: Models and Algorithms **ECE 2019 Spring 2018**
Prof Vwani Roychowdhury TA: Tianyi Wang
UCLA, Department of ECE Behnam Shahbazi

Project 3: Collaborative Filtering

Due Monday May 21, 2018 by 11:59 pm

Team Members

Jennifer MacDonald, UID: 604501712
Nguyen Nguyen, UID: 004870721
Sam Yang, UID: 604034791

1 Introduction

“Top Movie Picks” and “Frequently Bought Together” are words oftenly seen on Netflix and Amazon respectively. These are recommender systems (a subclass of information filtering system) that provide prediction of ratings or “preferences” to a user basing on his or her use. Recommender systems have become popular recently and are widely utilized in areas including movies, music, news, books, and products in general. For example, recommender systems such in Netflix take users’ feedback of “like” or “dislike” ratings for movies, and use it to infer how similar products would be viewed. Systems such in Amazon look at what items the user has bought previously and compare those against what is on the catalog. Amazon calls this system ‘item to item collaborative filtering’ which involves making recommendations at product level rather than user level.

In a recommender system, the entity that provides the feedback is referred to as the user, and the product being recommended is called the item. There are two basic types of recommender systems: user-item interactions, collaborative filtering methods, in which items are rated by users; and attribute information, content based methods, such as keywords about either users or items. In this project, we explored how we can utilize collaborative filtering to build a recommendation system in order to predict the ratings of the movies in the MovieLens dataset. Since the dataset also contains movie genre information, we can only utilize movie rating information.

2 Collaborative filtering models

Collaborative filtering, sometimes referred as social filtering, filters out information by using recommendations from users. In this project, our collaborative filtering system filters out movies using the provided ratings. To simply put, it is based on an idea that users agreed on their evaluation in the past are most likely will agree again in the future. A user who wants to see a movie, for example, might ask some friends for some movie recommendations. Thus, the recommendations from the friends who have similar interest are more likely trusted than strangers. Therefore, collaborative filtering methods aggregate large amounts of data in order to make predictions. Another example, in the case of Netflix, users' ratings of movies are used to make predictions, either basing on other users' ratings on the same movie, or on how one user will rate other movies. The underline challenge, however, is that users typically don't rate a wide variety of movies, and there are many movies that don't get rated. Therefore, the aggregated matrix of all the ratings can be sparse. Though, these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items.

Collaborative filtering is a useful method for these types of scenarios because it can predict how a movie will be rated based on the user's rating, or how others have rated it. If two users tend to have similar ratings for movies, then we can use one user's rating to predict the rating for that other user for the same movie. In this project, we implemented and analyzed the performance of two types of collaborative filtering methods:

- Neighborhood-based collaborative filtering
- Model-based collaborative filtering

Most collaborative filtering systems apply the neighborhood-based technique. In this approach, a number of users is selected based on their similarity to active user. Then, a prediction for the active user is made by calculating a weighted average of the ratings of the selected users. In this case, the weight given to a person's ratings is determined by the correlation between that person and the person for whom to make a prediction. The Pearson correlation similarity is typically used as defined below.

$$\mathrm{simil}(x,y) = \frac{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2} \sqrt{\sum\limits_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Where I_{xy} is the set of items rated by both user x and user y. However, the cosine-based similarity approach to determine cosine-similarity between two users can also be used which is defined below.

$$\text{simil}(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum\limits_{i\in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum\limits_{i\in I_x} r_{x,i}^2}\sqrt{\sum\limits_{i\in I_y} r_{y,i}^2}}$$

3 MovieLens dataset

This project uses the information in the MovieLens dataset given by the link http://files.grouplens.org/datasets/movielens/ml-latest-small.zip to build a recommendation system for the ratings of thousands of movies from hundreds of users using the movie rating information. The data can be represented as a two-dimensional matrix m x n, where the rows represent the users and the columns represent individual movies.

Question 1: Compute the sparsity of the movie rating dataset, where sparsity is defined by equation 1

Sparsity is defined as the number of ratings that are available divided by all possible ratings (if every user rated every movie) as defined by the following equation below. It is to measure how sparse the matrix is.

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

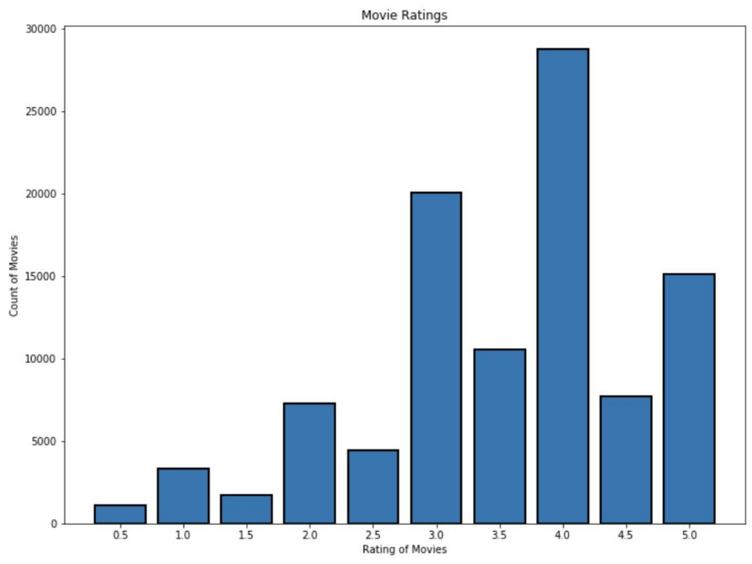
Sparsity: 0.0164

The sparsity of this matrix is extremely low, at just above 1% of the matrix. This means that the vast majority of ratings were not provided. This seems to be common since there are much more movies than users, thus it’s unlikely for all users to watch most of the movies and give rating on those movies.

Question 2: Plot a histogram showing the frequency of the rating values. To be specific, bin the rating values into intervals of width 0.5 and use the binned rating values as the horizontal axis. Count the number of entries in the ratings matrix R with rating values in the binned intervals and use this count as the vertical axis. Briefly comment on the shape of the histogram

Below is a histogram showing the frequency of number of ratings. We counted the number of ratings in each category by binning values within 0.5-width increments. Then, we plotted the values in the histogram, with the ratings of movies on the horizontal axis and the count of movies for each rating bin on the vertical axis as shown in Figure 1.

Figure 1: Histogram of Movie Ratings

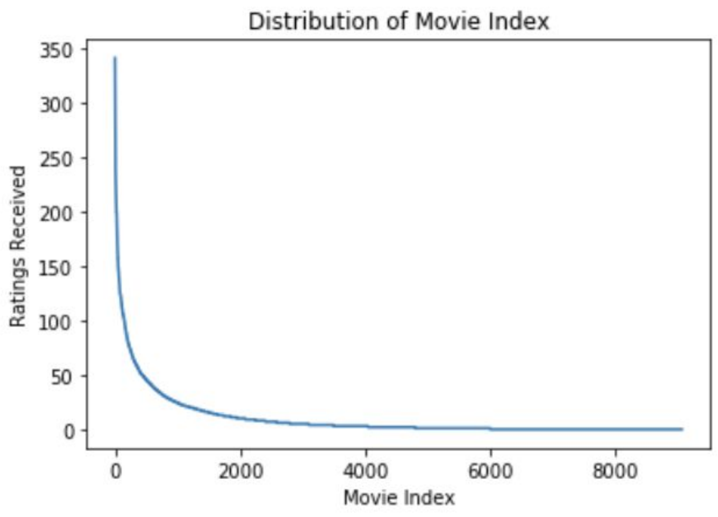


The histogram is not monotonic, but there is a general increase in the ratings of movies, with 4 being the most popular movie rating. Interestingly, the histogram shows that it is skewed to the left, which means that the users tend leave more positive feedbacks to the movies they liked and less negative feedbacks to movies they don't like. Though we think this seems to be a psychological behavior since most users don't like to leave negative feedbacks. There also seems to be a trend to only rate movies in whole numbers rather than in a number-and-a-half.

Question 3: Plot the distribution of ratings among movies. To be specific, the X-axis should be the movie index ordered by decreasing frequency and the Y -axis should be the number of ratings the movie has received.

Below is the distribution of the total ratings received to a particular movie index (each movie index is a distinct movie). We disregarded any users that did not provide any rating to the movie sine they would not provide much information. The movie index is ordered by decreasing frequency. Highest rating frequencies occur for some movies but rapidly drop. This parabolic curve means that there are only a few of movies that are popular which are rated by many users, while the unpopular movies only received less than 50 total ratings.

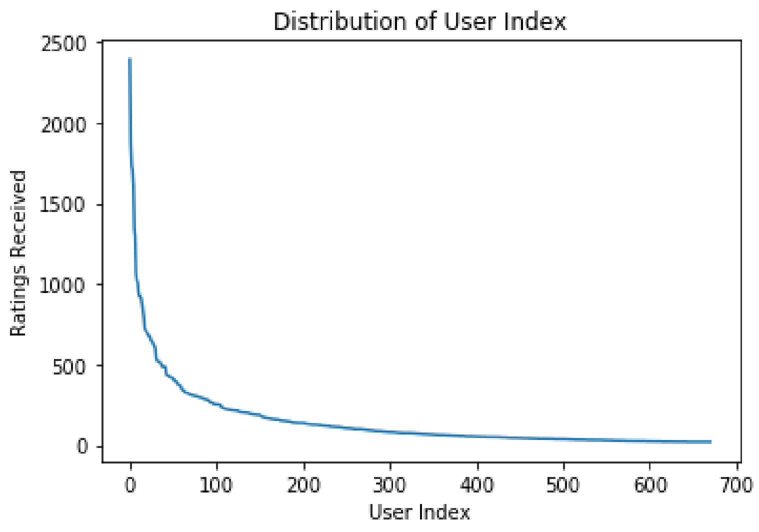
Figure 2: Distribution of Ratings among Movies



Question 4: Plot the distribution of ratings among users. To be specific, the X-axis should be the user index ordered by decreasing frequency and the Y -axis should be the number of movies the user have rated.

Below is the distribution of total ratings received to a particular user (each user index is a distinct user). Again, we disregarded any movies that have no rating since they would not provide much information, and ordered user index in decreasing order. We observed that most ratings were made by only a few users, but the majority of the users only provided less than 500 ratings.

Figure 3: Distribution of Ratings among Users



Question 5: Explain the salient features of the distribution found in question 3 and their implications for the recommendation process.

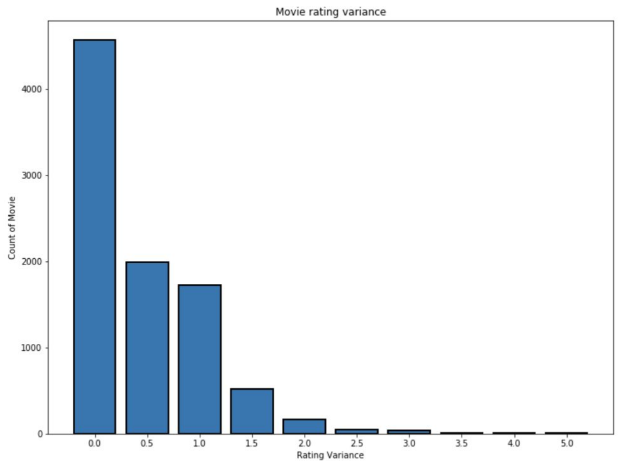
In the distribution found in question 3 above, we observed that only a few movies have the highest total ratings given, which means that there are few popular movies. This makes sense because popular movies are often watched by many users, hence that's why they're popular, and therefore are rated the most by the users. It is easy for a recommendation system to collect the ratings and predict these movies to other users.

Question 6: Compute the variance of the rating values received by each movie. Then, bin the variance values into intervals of width 0.5 and use the binned variance values as the horizontal axis. Count the number of movies with variance values in the binned intervals and use this count as the vertical axis. Briefly comment on the shape of the histogram

Figure 4 below is the variance distribution of the rating values received by each movie. Again we disregarded any movies with no ratings. The variance however is in increasing order. From result, we observed that the variance is around 1-1.5, meaning that most users rates a movie within a difference of 1 or at most 2.

To compute the variance of the rating values for each movie, we created a pivot table where all the unrated values were transformed to NaN values, then use np.var to calculate the variance among the columns, and bin the count of values in increments of 0.5. This histogram is not monotonic either, but it is strongly skewed to the right, which means that there tends to be a low variance in the distribution of ratings. This histogram exhibits that users tend to give ratings in their 'ideal' range and do rate movies that they haven't watched or liked, which is consistent with what we have learned before about mostly movies being rated highly.

Figure 4: Distribution of Variance of Rating Values



4 Neighborhood-based collaborative filtering

Now we're going to further discuss the use of neighborhood-based collaborative filtering to make predictions using the MovieLens dataset for user-user similarities or item-item similarities. There are two basic principles used in neighborhood-based models: user-user based models, and item-item based models. User-user based models are which similar users tend to rate movies in a similar fashion, so that one person's rating for a movie can be used for another person with similar tastes. On the other hand, item-item based models are when items are rated in a similar way, so a rating from a user for one movie can be applied to another movie that has been rated similarly. This project only focuses on the user-based model to implement collaborative filtering.

4.1 User-based neighborhood models

User-based neighborhood models are when an algorithm is developed to find users that are similar to the original user in terms of how they rate items. This is done by computing the similarity of the original user to all other users being considered. For this project, we use the Pearson-correlation coefficient to calculate the similarity.

4.2 Pearson-correlation coefficient

The Pearson-correlation coefficient, which calculates the similarities between two users, can be represented by a set of variables:
I_u : Item indices rated by a user u
I_v : Item indices rated by a user v
μ_u: Mean rating for a user u of r ratings
u_k: Rating of user u for a specific item k

Question 7: Write down the formula for μ_u in terms of I_u and r_{uk}

The mean rating for a user u with ratings can be calculated by taking the sum of ratings for the user that is in the set of item indices rated by that user.

$$\frac{\Sigma_{k \in Iu} r_{uk}}{\Sigma_{k \in Iu} 1}$$

Question 8: In plain words, explain the meaning of $I_u \cap I_v$. Can $I_u \cap I_v = \varnothing$
(Hint: Rating matrix R is sparse).

This represents the set of all items that were rated by both user u and user v. In terms of movies, they are the movies that both users rated. $I_u \cap I_v = \varnothing$ is when user u has never rated anything user v has rated, and if user v has never rated anything user u has rated. For example, if user u only rated movies created before 1970, and user v only rated movies created after 1970, then $I_u \cap I_v = \varnothing$.

All this fits together to calculate the Pearson-correlation coefficient, below:

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

4.3 k-Nearest neighborhood (k-NN)

Previously, we defined the similarity matrix. We now can define the Pearson-correlation coefficient to find the neighborhood of users. The k-nearest neighbor of a given user can be represented by P_u, which is the top k user with highest Pearson-correlation to user u.

4.4 Prediction function

Therefore, we can use the prediction function for the a rating with similar users, r_{uj} as seen below.

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

Question 9: Can you explain the reason behind mean-centering the raw ratings ($r_{vj} - \mu_v$) in the prediction function? (Hint: Consider users who either rate all items highly or rate all items poorly and the impact of these users on the prediction function)

Mean centering the movies helps to remove potential biases in users' rating styles. If user u is highly critical and rates most movies between 1-3, while user v is typically generous and rates movies between 3-5, yet they have the same movie preferences, the similarity in movie preferences might be lost due to the absolute differences in the ratings. In this case, it might be possible that user w with a generous rating style (between 3 and 5) but different movie preferences might be considered more similar to user u since their scores have more overlap. By mean-centering all the ratings, this helps remove some of these biases that arise from rating styles (highly critical or overly generous).

4.5 k-NN collaborative filter

Although we have calculated the formula for the neighborhood of users, there are built-in prediction functions in python that we used. We will use the built-in k-NN function in *surprise* library.

4.5.1 Design and test via cross validation

We had to design a k-NN collaborative filtering and test how well it performed using a 10-fold cross validation. X-fold cross-validation is when the data is partitioned equally into an X number of equal parts. All subsets except one are used to train the data, with the remaining one is used for testing. This process is repeated 10 times, with each of the 10 subsets used exactly once as the validation data.

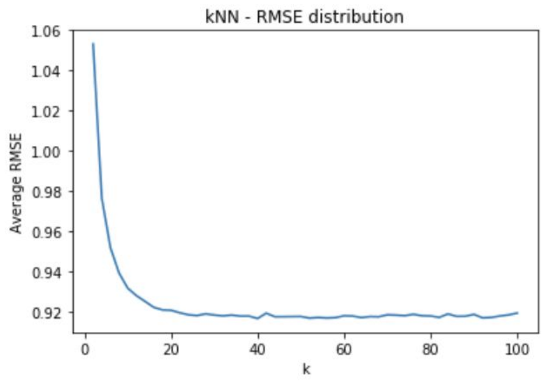
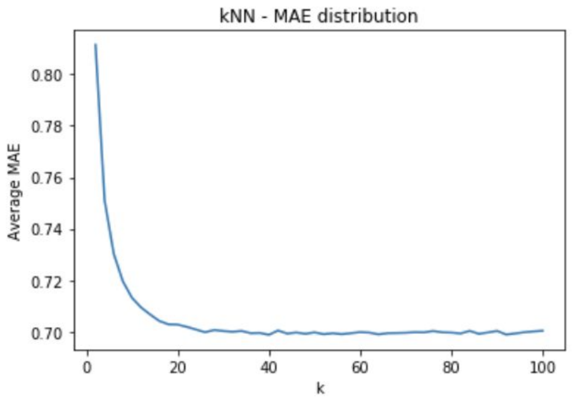
Question 10: Design a k-NN collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis) and average MAE (Y-axis) against k (X-axis).

We used the Pearson correlation equation from the surprise Python scikit library as listed below:

- http://surprise.readthedocs.io/en/stable/knn_inspired.html
- http://surprise.readthedocs.io/en/stable/model_selection.html#surprise.model_selection.validation.cross_validate
- http://surprise.readthedocs.io/en/stable/similarities.html

First, we load the movie rating dataset using load_from_df and initialize two arrays for RMSE and MAE to capture the data. We also create an array of k values from 2 to 100 in step sizes of 2. Then, we used a for loop to go through all the k values, where for each value of k we create a k-NN collaborative filter using the kNNWithMeans function and then cross validate the how well it performed. The mean RMSE of the cross validation for that k value gets pushed into the RMSE array and the mena MAE gets pushed into the MAE array below in Table 1. We plotted the average for both RMSE and MAE and against k.

Table 1: Average RMSE and MAE Measures for k-Values Using a k-NN Collaborative Filter

RMSE	MAE
	

The average for both RMSE and MAE rapidly drop as the k-value rises, showing that the variance and error is lowered the higher that the value of k is. However, there seems to be a point of diminishing returns around a k-value of 20.

Question 11: Use the plot from question 10, to find a ’minimum k’. Note: The term ’minimum k’ in this context means that increasing k above the minimum value would not result in a significant decrease in average RMSE or average MAE. If you get the plot correct, then ’minimum k’ would correspond to the k value for which average RMSE and average MAE converges to a steady-state value. Please report the steady state values of average RMSE and average MAE.

To calculate the minimum k, we looked at the plot from question 10 to see where our RMSE and MAE average values converged. For both plots, it looks like the line flattened out at k = 20 (or slightly after that, possibly k = 22). We looked to the vertical axis to see what our average RMSE and average MAE values were at that k value, and it appeared that at that point the RMSE value was equal to 0.92 and the MAE value was equal to 0.70. After this threshold, RMSE and MAE seems to reach a steady state. Higher k values after this point means longer computing time, but would not help much in filtering performance. These values can be seen below in Table 2.

Table 2: Minimum k and Corresponding Average RMSE and MAE From Plots in Table 1

Minimum k	20
Corresponding Average RMSE	0.92

Corresponding Average MAE	0.70

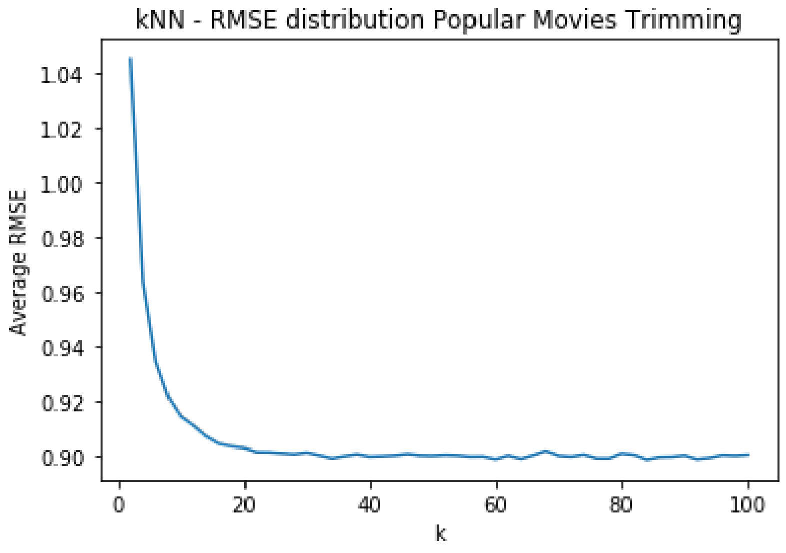
4.6 Filter performance on trimmed test set

We also make predictions using variations of the original data test set, where the data is trimmed to certain specifications. There are three ways that we trimmed the test set for this project: popular movie trimming, in which the movies that are 2 or less ratings are removed from the dataset; unpopular movie trimming which we keep movies that have received 2 or less ratings and remove everything else; and high variance movie trimming, in which the data test set is trimmed to only contain movies that have a variance of 2 or greater and have at least 5 ratings. We used the k-NN filter to predict movie ratings for these trimmed test sets.

Question 12: Design a k-NN collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate its performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the k-NN collaborative filter, however, we apply the trimming process on each test set (there are 10 test sets total because of 10-fold validations). Here, We trimmed the popular movie test set which removed movies that have more than total of 2 ratings. The results can be seen in Figure 5 below.

Figure 5: Average RMSE Measures for k-Values Using a k-NN Collaborative Filter with the Popular Movie Trimmed Test Set



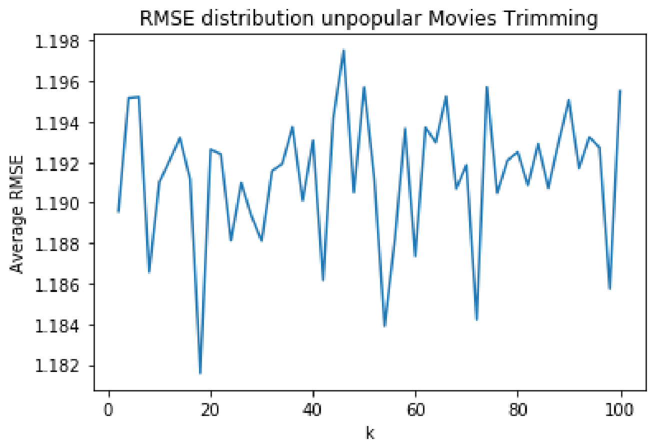
Minimum average RMSE: 0.8984

Similarly to the original data test set, the average RMSE value drops rapidly as the k-value rises, and it appears that the RMSE value starts to level out slightly after a k value increases. Though, the minimum average RMSE value we observed is 0.8984. When k is small, performance improves as k gets larger. However, the performance does not seem to increase much when k goes beyond the threshold as mentioned above. This means that in popular movie trimming test set (movies that have received more than 2 ratings are trimmed), we can definitely improve filtering performance as k increases (to a certain point) because we can minimize the bias effect caused by higher rating frequency movies.

Question 13: Design a k-NN collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the k-NN collaborative filter using the unpopular movie trimmed test set (movies that have received less than 2 ratings). The results can be seen in Figure 6 below. The minimum average RMSE we observed is 1.1816. However, the result does not follow the same trend in popular trimmed test set previously since increasing k values does not help with performance at all. This is expected because we only have a few popular movies to begin with in the original dataset. By removing movies that receiving less than 2 ratings, we're effectively truncating the test set to only popular movies. Since the testset has less movies now, it can contribute to large error when predicting a movie rating. For example, assume that the unpopular movie has a rating of 1 and 5. If the prediction is 2, although it is close to 1, it would still cause a high RMSE because it's far away from 5; same case if the prediction is 4. Furthermore, if the prediction is at 3, which is midpoint, it would still cause a high RMSE because now it's not close to both either actual ratings. Therefore improving k values does not seem to help with performance.

Figure 6: Average RMSE Measures for k-Values Using a k-NN Collaborative Filter with the Unpopular Movie Trimmed Test Set

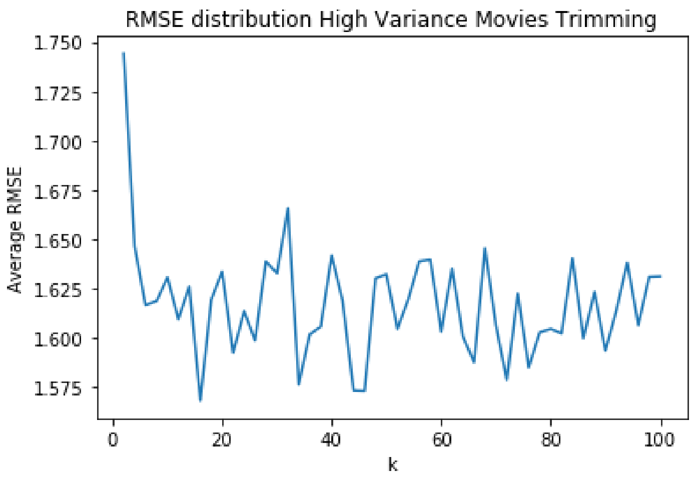


Minimum average RMSE: 1.1816

Question 14: Design a k-NN collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

Here, we repeat the same process above with the k-NN collaborative filter using the high variance movie trimmed test set (movies that have variance of the ratings of between 2 and 5 ratings in the entire dataset). The results can be seen in Figure 7 below.

Figure 7: Average RMSE Measures for k-Values Using a k-NN Collaborative Filter with the High Variance Movie Trimmed Test Set



Minimum average RMSE: 1.5683

In Figure 7, RMSE decreases rapidly but it reaches a somewhat steady-state (still with a lot of variance, comparatively) much earlier where k > 5. Anything beyond this would not help much in terms of performance. Because our testset contains high variance data, there will always be some kind of error. Therefore, higher k values would not help much in this case. The steady state seems to hover between 1.575 and 1.650. Therefore, we can conclude that increase k values does help RMSE in high variance movie trimming data set, but the k values would be smaller.

4.6.1 Performance evaluation using ROC curve

The ROC curve is used to visualize how well binary classifiers have performed by plotting the true positive rate against the false positive rate. For the problems in this project, it graphs the accuracy of the recommended movies when converted to a binary output (instead of the continuous output we had been using up until now). To convert to a binary rating, we counted anything that was greater than a threshold as 1, that they would recommend the item, and anything less would be an output of 0, meaning they would not recommend the item.

Question 15: Plot the ROC curves for the k-NN collaborative filter designed in question 10 for threshold values [2.5, 3, 3.5, 4]. For the ROC plotting use the k found in question 11. For each of the plots, also report the area under the curve (AUC) value.

With this particular problem, we had to split the dataset, with 90% for training and 10% for testing, and then using Python sklearn's ROC plotting function (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html), plotted the ROC curves for the k-NN collaborative filter. We created three arrays for the false positive rate (FPR), the true positive rate (TPR), and and the area under the curve (AUC). Then, for each of the thresholds, we create a basic k-NN filtering algorithm using the minimum RMSE value as k, and then fit and predict the data as normal. We stepped through the predicted outcomes and appended a 1 if it was above the threshold and a 0 if it was below. We then call the ROC curve and add the FPR, the TPR, and the AUC to each corresponding array. After that, we plot the ROC curve based on these figures, shown in Figure 8.

Basing on the ROC curve with the threshold values: 2.5, 3, 3.5, 4, The largest AUC is 0.7757 where the threshold is 3. The larger AUC indicates that there is more true positive rate than false negative rates.

Figure 8: ROC Curves for the k-NN Collaborative Filter for the Given Thresholds

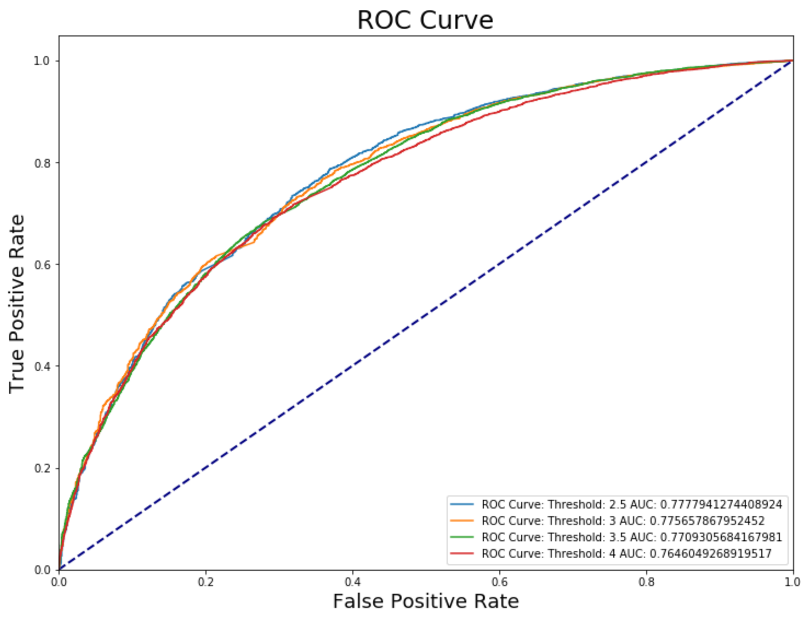


Table 3: Area Under the Curve For Different Thresholds Using the k-NN Collaborative Filter

Threshold	Area Under the Curve
2.5	0.7778
3	0.7757
3.5	0.7709
4	0.7646

5 Model-based collaborative filtering

Next, we moved onto model-based collaborative filtering, where predictions are made for users' items that have not been rated. There are many different implementations of this, like decision trees and bayesian methods, but for this project we will be using latent factor based models for collaborative filtering.

5.1 Latent factor based collaborative filtering

Latent factor collaborative filtering differ from what we had done before such that it fills out all missing entries in the matrix to predict ratings of items for each user. This can be done since there tends to be a high correlation between parts of rows and columns, so that the predicted matrix of user-item relationships can be represented by a low-rank matrix. This method, known as matrix factorization can be represented by

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

where m and n are the matrix dimensions, k is the number of latent factors, and the goal is to find U (a m x k matrix) and V (a n x k matrix) by calculating using known values in the original sparse matrix. These values, represented as W, is calculated by

$$W_{ij} = \begin{cases} 1, r_{ij} \text{ is known} \\ 0, r_{ij} \text{ is unknown} \end{cases}$$

and modifies the original formula slightly to

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

Due to the nature of drawing from a sparse matrix, the set of ratings tends to be small, which can cause an issue of over-fitting. To fix this, regularization can be added using the regularization parameter λ, which is always non-negative and acts as a weight for the regularization term. The formula, below, is an expanded version of the original with the regularization and weights:

$$\underset{U,V}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2$$

We looked at two different versions of this matrix factorization, the non-negative matrix factorization (NNMF) and the matrix factorization with with bias (MF with bias).

5.2 Non-negative matrix factorization (NNMF)

If a ratings matrix is non-negative, then non-negative matrix factorization can be used. Again, we modify our latent factor based collaborative filtering algorithm to take this into account, as seen below:

$$\begin{aligned} &\underset{U,V}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2 \\ &\text{subject to} \quad U \geq 0, V \geq 0 \end{aligned}$$

Optimization algorithms like stochastic gradient descent (SGD) and alternating least-squares (ALS). There are a few differences between these algorithms: while SGD has a high initial sensitivity and step size, ALS is less sensitive and converges faster than SGD. This can happen because while ALS is solving for either the U or V matrix, it keeps the one not being solved as fixed. Although ALS can often be a better-suited algorithm for some problems, we used SGD in this project because the python package we use contains functions to perform SGD. The end result would have ended up the same even if we implemented ALS because both converge on the MovieLens dataset.

Question 16: Is the optimization problem given by equation 5 convex? Consider the optimization problem given by equation 5. For U fixed, formulate it as a least-squares problem.
The optimization problem, shown below, can't be convex. The reason is that this function is a non-constant function with more than one global minima.

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

To turn this into a least-squares problem for U fixed, we must translate the problem to solve for V. For example, a matrix X with target value y could have a regularized least squares optimization problem represented as $w = (X^TX + \lambda I)^{-1}X^Ty$, where X^T is the transposition of the matrix X. If you swap out U for X and R for the target y, it turns into $V = (UU^T+\lambda I)^{-1}UR$.

5.2.1 Prediction function

After solving the optimization problem, we can use the optimized matrices I and V for the missing ratings using this formula:

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

Although we can use all these equations to build our own collaborative filter, there is a built-in one in surprise.

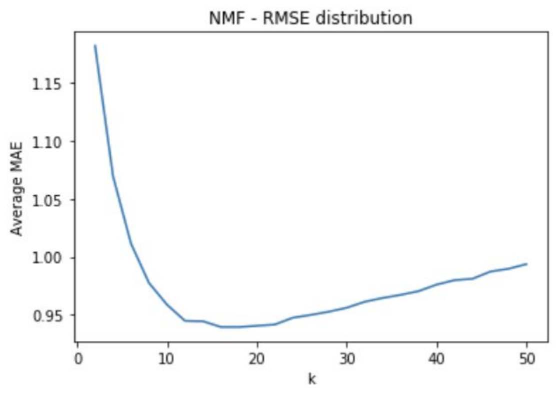
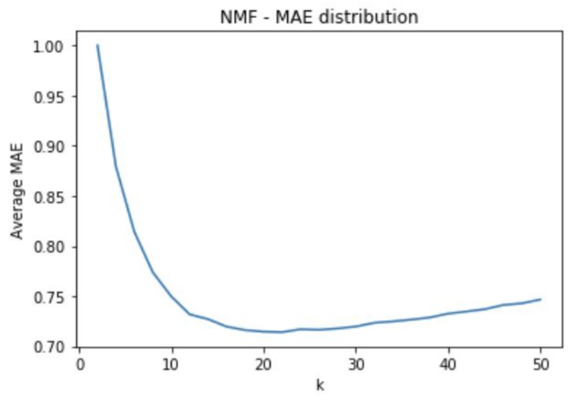
5.2.2 Design and test via cross-validation

Now, we moved on to finally creating a NNMF-based collaborative filter, and again, tested its performance with the 10-fold cross validation.

Question 17: Design a NNMF-based collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross-validation. Sweep k *(number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. Plot the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis). For solving this question, use the default value for the regularization parameter.*

Using surprises built-in NNMF-based collaborative filter (http://surprise.readthedocs.io/en/stable/matrix_factorization.html), we created a collaborative filter in the same method that we had created the k-NN collaborative filter: We created RMSE and MAE arrays and looped through for each iteration of k (number of latent factors), creating the NNMF-based collaborative filter using the built-in functions in surprise, and appending the average RMSE and MAE values. Our range of number of latent factors is from 2 to 50 in step sizes of 2. Each k computes the average RMSE and MAE obtained from all 10 folds.

Table 4: Average RMSE and MAE Measures for k-Values Using a NNMF Collaborative Filter

RMSE	MAE
	

Both RMSE and MAE distribution exhibit a dramatic decrease as k values increase, as seen in Table 4 above. Interestingly, both RMSE and MAE show a slow increase behavior as k increases when k > 20 (about). We suspect that because NMF is a random, non-unique algorithm that can converge prematurely at local minima. Thus, as k values increase, it's feasible that NMF converge prematurely for higher k values.

Question 18: Use the plot from question 17, to find the optimal number of latent factors. Optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. Please report the minimum average RMSE and MAE. Is the optimal number of latent factors same as the number of movie genres?

Table 5: Optimal Number of Latent Factors and Corresponding Average RMSE and MAE

	RMSE	MAE
Optimal Number of Latent Factors	18	22
Corresponding Average	0.9394	0.7140

To get the number of genres, we read the movies.csv file, delimiting with a comma. We then joined the all strings in the genre column, and split again using the vertical bar as a delimiter and wherever there was a capital letter. Since the number of genres is 19, the optimal number of latent factors is one off from the number of genres. This shows that RMSE is a closer estimation of the optimal number of latent factors.

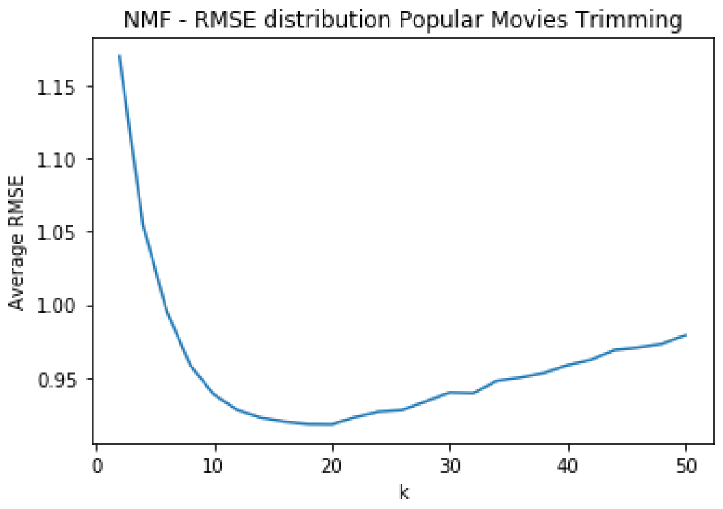
5.2.3 NNMF filter performance on trimmed test set

We then tested how well the NNMF filter performed with the trimmed test sets that we had previously created.

Question 19: Design a NNMF collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the NNMF collaborative filter using the popular movie trimmed test set which removed movies that have more than 2 ratings. We sweeped k (number of latent factors) from 2 to 50 in step sizes of 2. The results can be seen in Figure 9 below. The minimum average RMSE we observed is 0.9178.

Figure 9: Average RMSE Measures for k-Values Using a NNMF Collaborative Filter with the Popular Movie Trimmed Test Set



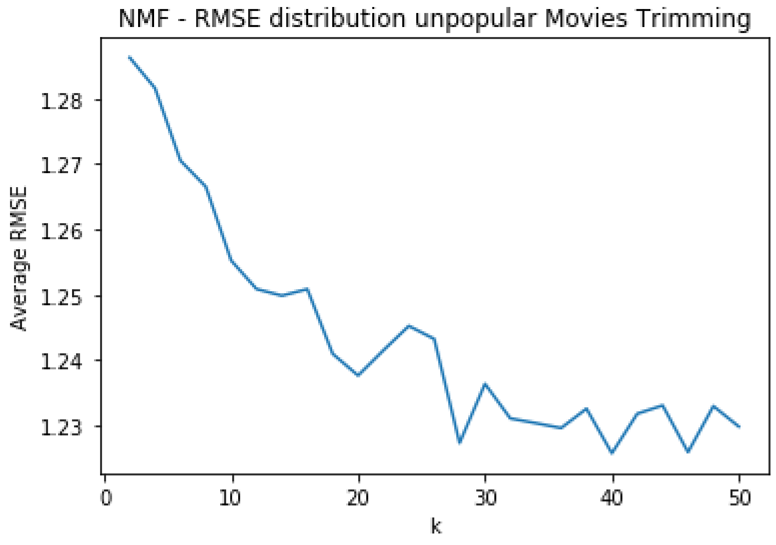
Minimum average RMSE: 0.9178

Figure 9 exhibits the same trend which RMSE increases when k > 20. This makes sense since popular movie trimmed test set exhibits similar trend with its original dataset, consistent with the kNN collaborative filtering above.

Question 20: Design a NNMF collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeated the same process above with the NNMF collaborative filter using the unpopular movie trimmed test set. The results can be seen in Figure 10 below. The minimum RMSE we observed is 1.2257.

Figure 10: Average RMSE Measures for k-Values Using a NNMF Collaborative Filter with the Unpopular Movie Trimmed Test Set



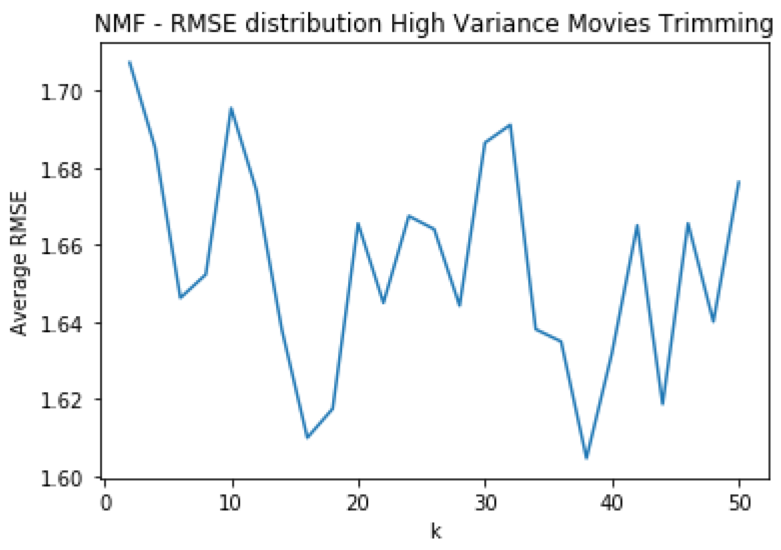
Minimum average RMSE: 1.2257

Interestingly, as k increases, average RMSE decreases which is the opposite of what we have been observing for other unpopular movie trimmed test sets.

Question 21: Design a NNMF collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE

We repeat the same process above with the NNMF collaborative filter using the high variance movie trimmed test set. The results can be seen in Figure 11 below. The min average RMSE we observed is 1.6047 which occurs when k = 18. However, we can’t say that this is optimized k value since it seems to exhibit constant fluctuation.

Figure 11: Average RMSE Measures for k-Values Using a NNMF Collaborative Filter with the High Variance Movie Trimmed Test Set



Minimum average RMSE: 1.6047

5.2.4 Performance evaluation using ROC curve

We again evaluated the performance of this filter using the ROC curve.

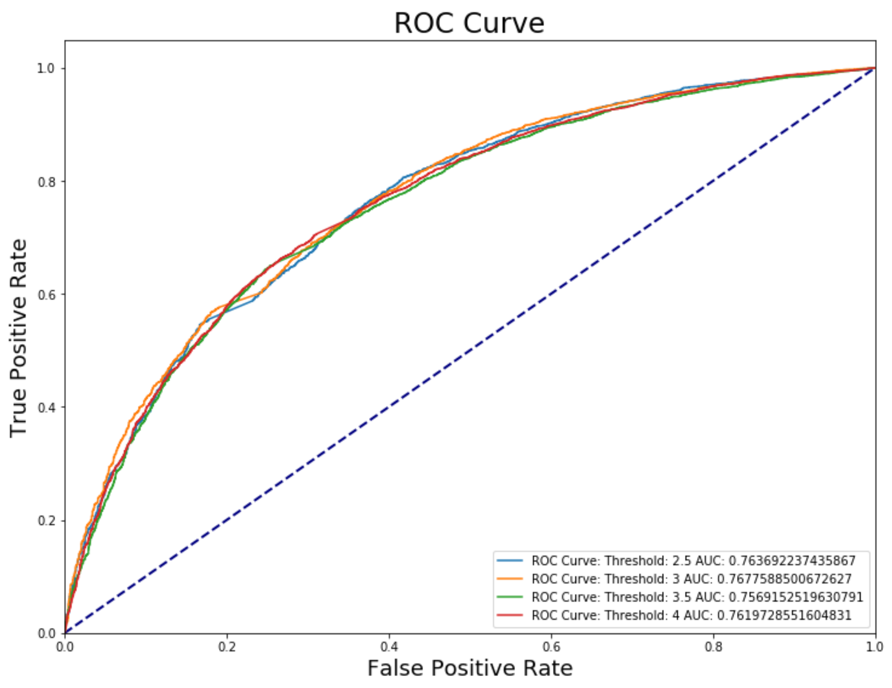
Question 22: Plot the ROC curves for the NNMF-based collaborative filter designed in question 17 for threshold values [2.5,3,3.5,4]. For the ROC plot- ting use the optimal number of latent factors found in question 18. For each of the plots, also report the area under the curve (AUC) value.

We used the same methodology for the NNMF-based collaborative filter ROC curve as the ROC curve with the k-NN filter as seen in Figure 12 below. Basing on the ROC curve, we observed that Threshold 3 returns the highest AUC score which is 0.7678.

Table 6: Area Under the Curve For Different Thresholds Using the NNMF Collaborative Filter

Threshold	Area Under the Curve
2.5	0.7637
3	0.7678
3.5	0.7569
4	0.7619

Figure 12: ROC Curves for the NNMF Collaborative Filter for the Given Thresholds



5.2.5 Interpretability of NNMF

There is one big advantage of using NNMF as a collaborative filter over other matrix factorization algorithms: it is very each to interpret the results. We explored this further in the following problems.

Question 23: Perform Non-negative matrix factorization on the ratings matrix R to obtain the factor matrices U and V , where U represents the user-latent factors interaction and V represents the movie-latent factors interaction (use k = 20). For each column of V , sort the movies in descending order and report the genres of the top 10 movies. Do the top 10 movies belong to a particular or a small collection of genre? Is there a connection between the latent factors and the movie genres?

After performing NNMF on the ratings matrix, the resulting factors in V can represent interest groups that the users fall into. By taking the most highly weighted movies in each column, we can get an idea of what “category” each latent factor is trying to capture. Since users typically are drawn to certain genres of movies, we can look at whether NNMF might illustrate how well genres define interest groups. Here were a couple of interest groups we identified:

Table 7: Top 10 Movies for Each Column of V

INTEREST GROUP 3	INTEREST GROUP 11	INTEREST GROUP 13
Action|Adventure	Horror	Adventure|Comedy|Thriller
Comedy	Crime|Drama|Thriller	Drama|Romance|War
Action|Adventure|Drama|Fantasy	Comedy	Comedy|Drama|Romance
Drama|Mystery	Horror	Documentary
Action|Crime|Drama|Thriller	Comedy|Sci-Fi	Comedy|Drama|Romance
Drama	Comedy	Horror
Drama|Romance	Documentary	Comedy|Romance
Action|Adventure|Sci-Fi	Horror|Sci-Fi	Comedy
Action|Adventure|Fantasy|	Horror|Sci-Fi	Comedy|Drama|Romance
Adventure|Children|Fantasy	Musical	Comedy|Drama

The NNMF latent factors seem to show how some of the interest groups can be related to genre. Interest group 3 is strongly tied to movies in the genre of action and adventure, interest group 11 captures horror, sci-fi, and comedy movies, and interest group 13 seems to highlight dramatic or romantic comedy movies.

To interpret this further, users with a preference for romantic comedies may have a high 13th user latent factor, since the dot product of the 20 user factors and the 20 movie factors should be high if the rating is to have a large value. If we wanted to for some reason identify rom-com lovers, we might then sort users by the magnitude of the value in the 13th column of the U matrix.

5.3 Matrix factorization with bias (MF with bias)

In this part of the project, we switched over to the other method of matrix factorization, the matrix factorization with bias. This modification adds bias for each user and item, seen below:

$$\underset{U,V,b_u,b_i}{\text{minimize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2 + \lambda \sum_{u=1}^{m} b_u^2 + \lambda \sum_{i=1}^{n} b_i^2$$

Where b represents the added bias modification. The b_u is the bias of the user and b_i is the bias of the item, and the algorithm tries to optimize these biases with matrices U and V.

5.3.1 Prediction function

Once the the variables have been optimized, then we can calculate the rating of a user for a movie, r_{ij}, where r_{ij} is:

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

For this formula, μ is the mean if the ratings, and b_i and bj are the biases for items i and j, respectively.

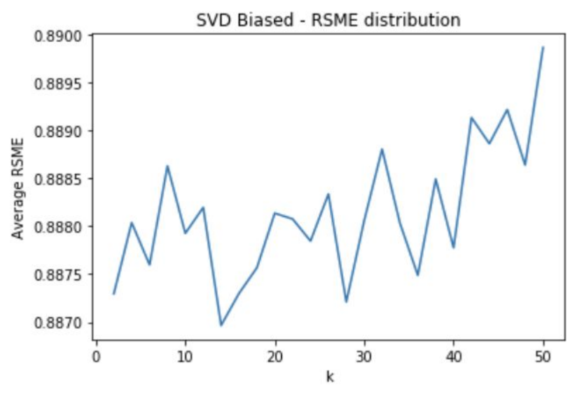
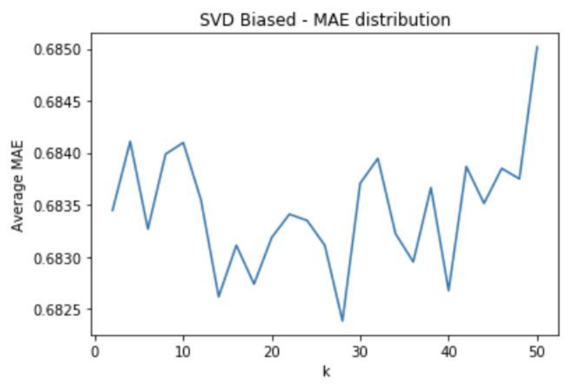
5.3.2 Design and test via cross-validation

We again designed and tested our matrix factorization with bias (MF)-based collaborative filter with the 10-fold cross validation.

Question 24: Design a MF with bias collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross-validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. Plot the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis). For solving this question, use the default value for the regularization parameter.

Using surprise library built-in SVD with bias parameter set to true, we created a model similar to that we had created the k-NN and NNMF collaborative filters. In previous report, we had mentioned that SVD does perform better than NMF since we learned that NMF does tend to converge to a local minima which was difficult to compare. Thus, we decided to use SVD instead of NMF for this biased model. Table 8 below is the average RMSE and MAE average values we observed.

Table 8: Average RMSE and MAE Measures for k-Values Using a MF with Bias Collaborative Filter

RMSE	MAE
	

There appears to be a consistency for both the average RMSE and MAE as k value increases, as seen in Table 8 above. Though, there is an upward trend but the difference in average RMSE is quite low that we can assume that it's a steady state with the current given specification. However, we would like to see how the model performs with a much wide range of k values.

Question 25: Use the plot from question 24, to find the optimal number of latent factors. Optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. Please report the minimum average RMSE and MAE.

To calculate the optimal number of latent factors, looked at the plot from question 24 to see where our RMSE and MAE average values were at the minimum. For the RMSE plot, it looked like the line was at its minimum at 13, and for the MAE plot, it looked like the line was at its minimum at around 28. We looked to the vertical axis to see what our average RMSE and average MAE values were at that k value, and it appeared that at that point the RMSE value was equal to 0.89 and the MAE value was equal to 0.68. The results are shown below in Table 9.

Table 9: Optimal Number of Latent Factors and Corresponding Average RMSE and MAE

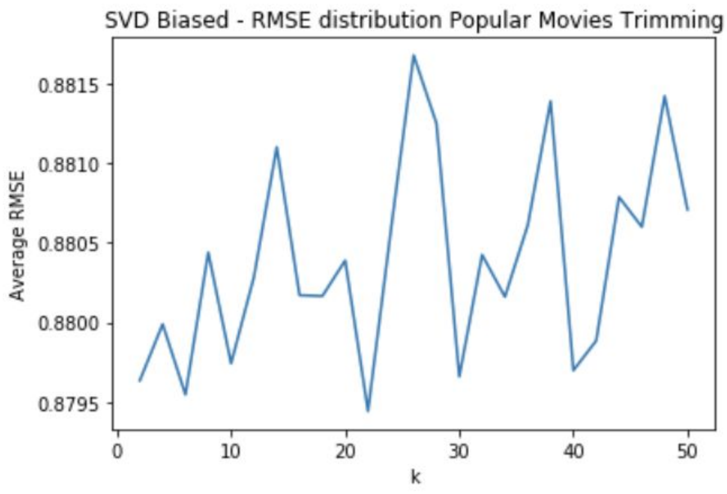
	RMSE	MAE
Optimal Number of Latent Factors	13	28
Corresponding Minimum Average	0.89	0.68

5.3.3 MF with bias filter performance on trimmed test set

Question 26: Design a MF with bias collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate its performance using 10-fold cross validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the MF with bias (using built-in SVD) collaborative filter using the popular movie trimmed test set which removed movies that have more than 2 ratings. The results can be seen in Figure 12 below.

Figure 12: Average RMSE Measures for k-Values Using a MF with Bias Collaborative Filter with the Popular Movie Trimmed Test Set



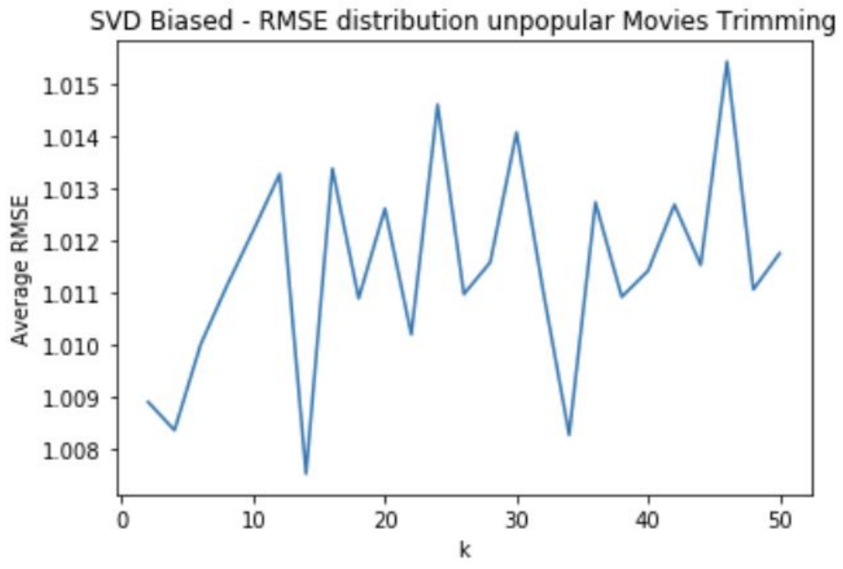
Minimum average RMSE: 0.8794

Again, the difference in average RMSE is quite low, with very small fluctuating variance as k increases. Since the graph is magnified, the changes are actually significantly smaller than it would appear.

Question 27: Design a MF with bias collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the MF (using built-in SVD) with bias collaborative filter using the unpopular movie trimmed test set. The results can be seen in Figure 13 below. Similarly, there is only a little more fluctuation using this trimmed test set.

Figure 13: Average RMSE Measures for k-Values Using a MF with Bias Collaborative Filter with the Unpopular Movie Trimmed Test Set

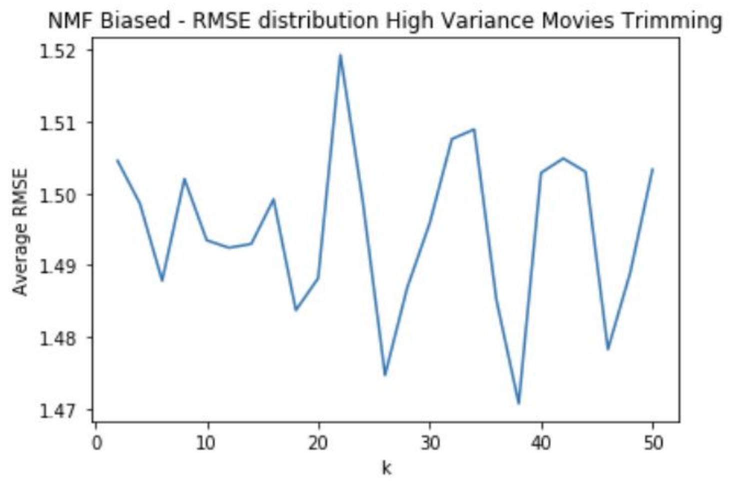


Minimum average RMSE: 1.0075

Question 28: Design a MF with bias collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate its performance using 10-fold cross validation.Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.

We repeat the same process above with the MF with bias collaborative filter using the high variance movie trimmed test set. The results can be seen in Figure 14 below. There is a considerable more variation with this trimmed test set, and as k increases, there seems to be a variable impact on how well the collaborative filter performs.

Figure 14: Average RMSE Measures for k-Values Using a MF with Bias Collaborative Filter with the High Variance Movie Trimmed Test Set



Minimum average RMSE: 1.4707

5.3.4 Performance evaluation using ROC curve

We again evaluated the performance of this filter using the ROC curve.

Question 29: Plot the ROC curves for the MF with bias collaborative filter designed in question 24 for threshold values [2.5,3,3.5,4]. For the ROC plotting use the optimal number of latent factors found in question 25. For each of the plots, also report the area under the curve (AUC) value.

We used the same methodology for the MF with bias collaborative filter ROC curve as the ROC curve with the k-NN and NNMF filters. The results can be seen below in Figure 15, with the AUC values in Table 10.

Figure 15: ROC Curves for the MF with Bias Collaborative Filter for the Given Thresholds

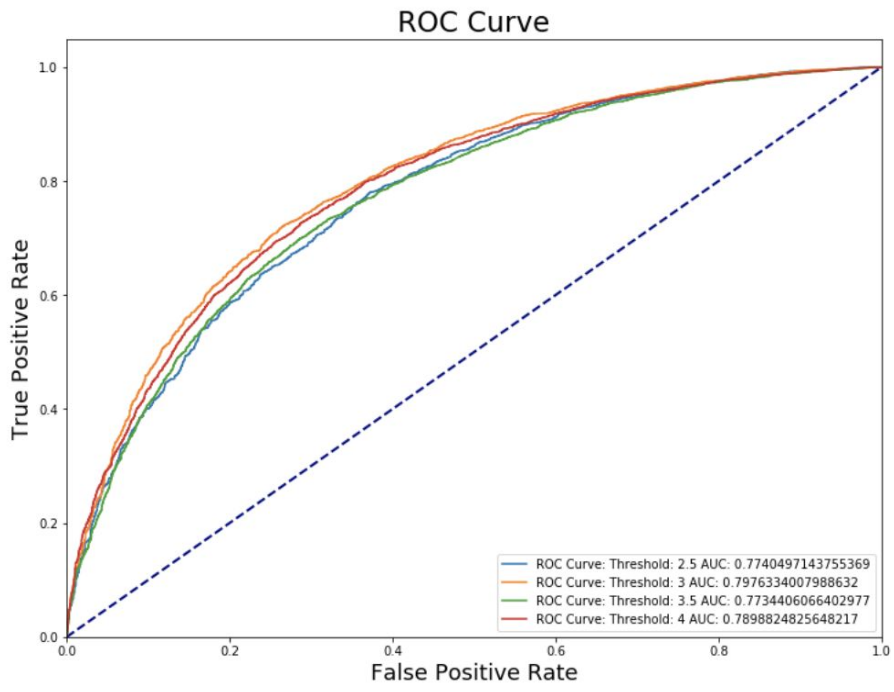


Table 10: Area Under the Curve For Different Thresholds Using the MF with Bias Collaborative Filter

Threshold	Area Under the Curve
2.5	0.7740
3	0.7976
3.5	0.7734
4	0.7899

Basing on the ROC curve with the threshold values: 2.5, 3, 3.5, 4. The largest AUC is 0.7976 where the threshold is 3. The larger AUC indicates that there is more true positive rate than false negative rates.

6 Naive collaborative filtering

We also used a naive collaborative filter to predict ratings in the dataset, which uses the mean rating of each user to predict movies that have not been rated.

6.1 Prediction function

Specifically, the prediction function would be represented as $r_{ij} = \mu_i$, where i is the user and j is the item, and μ_i is the mean rating out of the ratings for the user. The formula can be seen below.

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

6.2 Design and test via cross-validation

For the following questions, we will use the prediction function that we defined above and test how well the filter performed by testing with a 10-fold cross validation.

First, we built our own prediction algorithm with a fit prediction similar to the fit prediction in Python's sklearn.

Question 30: Design a naive collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

We called the surprise algorithm that we had built and used 10-fold cross validation to train and test the data in the original ratings dataset in MovieLens, taking the accuracy in each iteration. The iterations are summed together and then divided by the number of folds we used, which was 10. The results are shown in Table 11 below.

6.3 Naive collaborative filter performance on trimmed test set

Question 31: Design a naive collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate its performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

We repeated the same process using the popular movie trimmed test set. The results are shown in Table 11 below, where the average RMSE value is slightly lower than when using the original test set.

Question 32: Design a naive collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate its performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

We repeated the same process using the unpopular movie trimmed test set. The results are shown in Table 11 below where the average RMSE value is mildly higher than when using the original test set.

Question 33: Design a naive collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate its performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

We repeated the same process using the high variance movie trimmed test set. The results are shown in Table 11 below. There is a much higher average RMSE, as expected.

Table 11: Average RMSE Values Using a Naive Collaborative Filter

Test Set	**Average RMSE**
Original MovieLens Test Set	0.9624
Popular Movie Trimmed Test Set	0.9596
Unpopular Movie Trimmed Test Set	1.0141
High Variance Movie Trimmed Test Set	1.5207

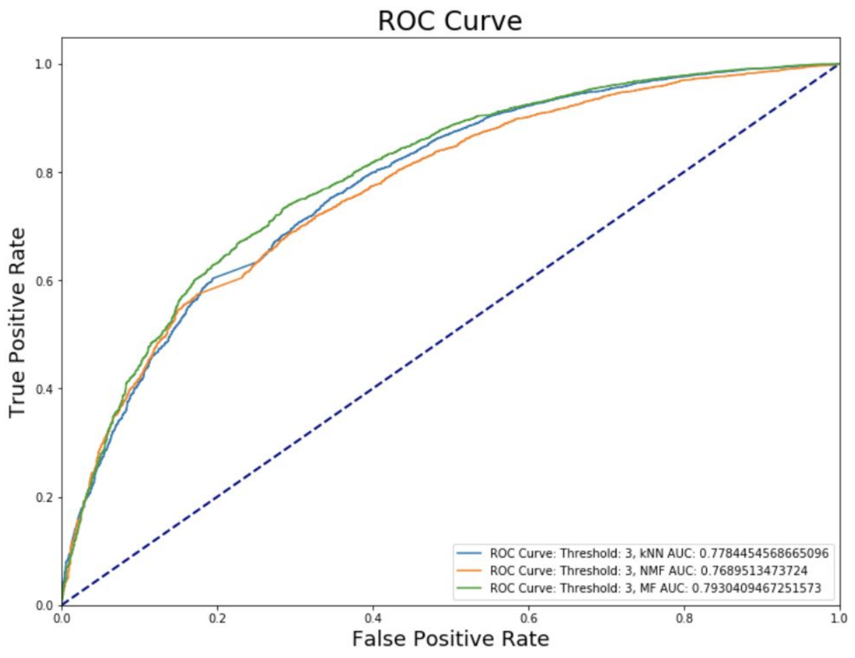
7 Performance comparison

After testing out several different collaborative filters, we also compared the performance between them.

Question 34: Plot the ROC curves (threshold = 3) for the k-NN, NNMF, and MF with bias based collaborative filters in the same figure. Use the figure to compare the performance of the filters in predicting the ratings of the movies.

We combined all the ROC curves for the three collaborative filters.

Figure 16: ROC Curves for the k-NN, NNMF, and MF with Bias Collaborative Filter



The curves in the plot in Figure, plus the actual calculated AUC for each filter, demonstrates that the MF with bias tended to perform better with the given data with an AUC value of 0.7930. The next best performance is from using the k-NN collaborative filter with an AUC value of 0.7784, and finally the NNMF collaborative filter gave an AUC value of 0.7690.

8 Ranking

There are two popular ways to go about formulating a recommendation problem: prediction version of problem, where it predicts the rating value, and ranking version of the problem, where it recommends a set of top k items for a user. At this point in the project, we looked at two ways to solve the ranking problem. Either create an algorithm to solve the ranking problem, like what we had been doing previously, or solve the problem and rank the predictions.

8.1 Ranking predictions

By solving the problem and then ranking the predictions, we can compute the predicted ratings using one of the collaborative filters and store the ratings. Then, we can sort the list in descending order. Lastly, we can select the top t-items from that list.

8.2 Evaluating ranking using precision-recall curve

A precision-recall curve is a tool to measure the performance of ranking predictions. The equation for precision and recall are:

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

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

where S(t) is a set of items that are recommended to a user, and G is a set of items that are actually liked by a user.

Question 35: Precision and Recall are defined by the mathematical expressions given by equations 12 and 13 respectively. Please explain the meaning of precision and recall in your own words.

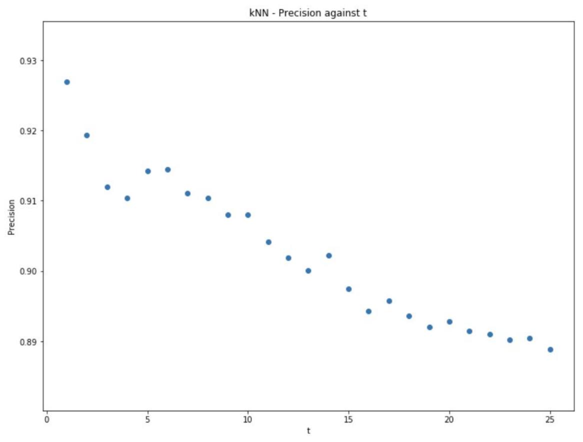
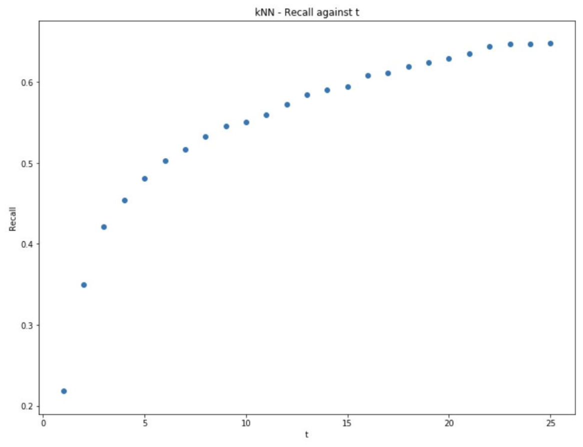
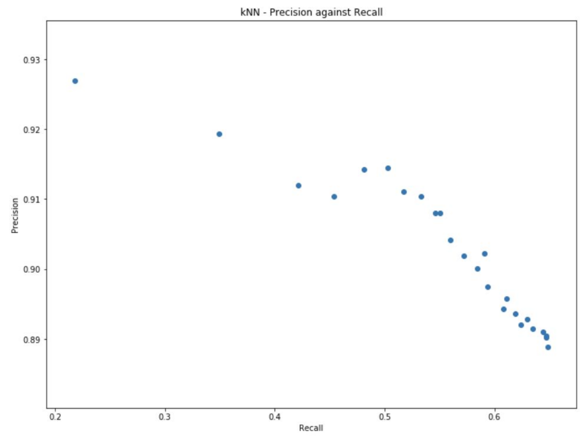
Precision looks at all of the predicted values from the model, and tells us which of those were correct. It informs us about how reliable a prediction is after the model deems it relevant.

Recall looks at all the ratings that the user actually likes, and within that set, how many overlap with the model's predictions. This gives us a sense of how much of the real values the model actually retrieves.

A model can have high precision and low recall if its really good at predicting true positives from a small subset of the actual positives. It can have low precision and high recall if it predicts everything to be positive while there are many non-positives in the data. It has high precision and high recall if it identifies all the positive values in the ground truth as positive.

Question 36: Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using k-NN collaborative filter predictions. Also, plot the average recall (Y-axis) against t (X-axis) and average precision (Y-axis) against average recall (X-axis). Use the k found in question 11 and sweep t from 1 to 25 in step sizes of 1. For each plot, briefly comment on the shape of the plot.

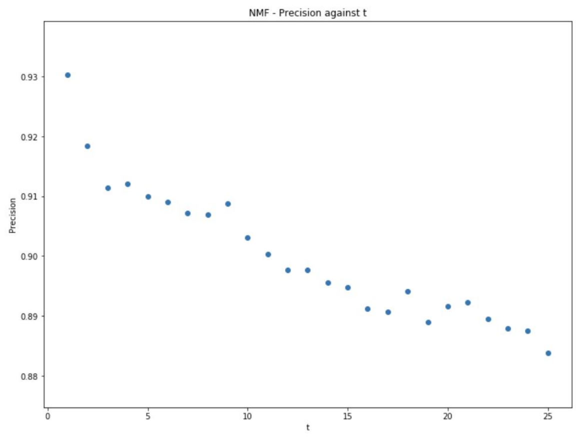
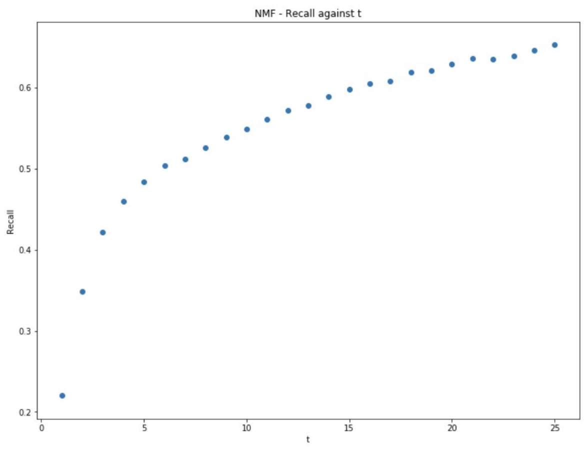
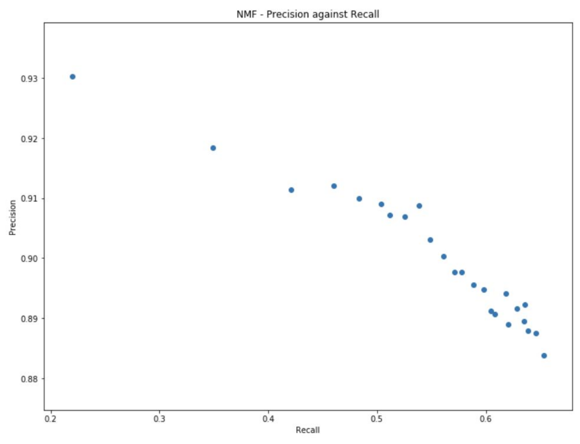
Table 12: Average Precision and Recall Using k-NN Collaborative Filter Predictions

KNN: precision vs. t The magnitude of average precisions all remain relatively high (between 0.89 and 0.94). There is a slight decrease as t increases. Intuitively this makes sense, as it is likely that the model's one highest predicted rating also happens to have a rating above 3.0 in reality. It is more difficult to maintain this consistently across, say 20 predicted ratings, which is why we might see a slight decrease over t.	
KNN: recall vs. t As expected, the recall increases with increasing t. It reaches a recall of about 0.7 with the t = 25 (highest t). For t = 1, the recall is 1 if the model guesses the highest predicted movie, and is 0 otherwise. Guessing the one favorite movie is intuitively more difficult than guessing a group of favorite movies, which may explain this increasing trend we observe	
KNN: precision vs. recall As explained earlier, it is intuitive that precision tends to decrease as t increases and that recall tends to increase as t increases. Since these are both shown in the first two plots, it also follows that precision should decrease as a function of increasing recall, as that is associated with increasing t.	

Question 37: Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using NNMF-based collaborative filter predictions. Also, plot the average recall (Y-axis) against t (X-axis) and average precision (Y-axis) against average recall (X-axis). Use optimal number of

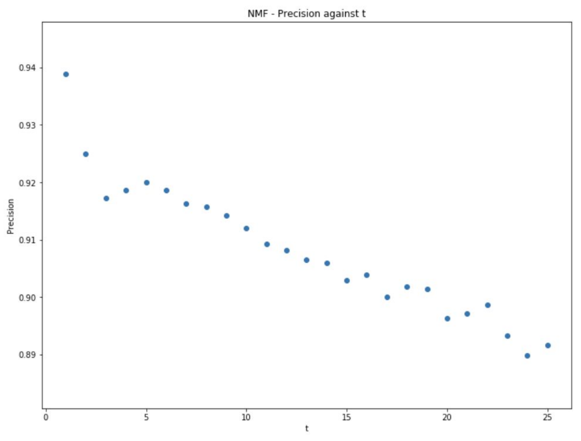
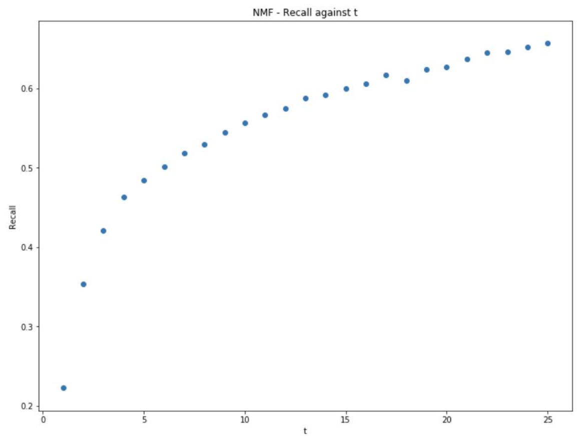
latent factors found in question 18 and sweep t from 1 to 25 in step sizes of 1. For each plot, briefly comment on the shape of the plot.

Table 13: Average Precision and Recall Using NNMF Collaborative Filter Predictions

NMF: precision vs. t The magnitude of average precisions all remain relatively high (between 0.89 and 0.93). There is a slight decrease as t increases. Intuitively this makes sense, as it is likely that the model's one highest predicted rating also happens to have a rating above 3.0 in reality. It is more difficult to maintain this consistently across, say 20 predicted ratings, which is why we might see a slight decrease over t.	
NMF: recall vs. t As expected, the recall increases with increasing t. It reaches a recall of about 0.7 with the t = 25 (highest t). For t = 1, the recall is 1 if the model guesses the highest predicted movie, and is 0 otherwise. Guessing the one favorite movie is intuitively more difficult than guessing a group of favorite movies, which may explain this increasing trend we observe	
NMF: precision vs. recall As explained earlier, it is intuitive that precision tends to decrease as t increases and that recall tends to increase as t increases. Since these are both shown in the first two plots, it also follows that precision should decrease as a function of increasing recall, as that is associated with increasing t.	

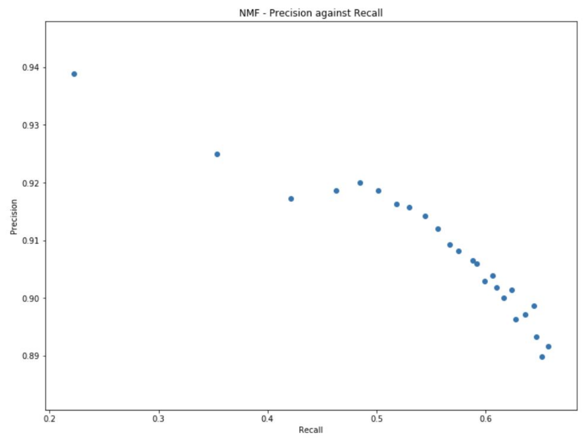
Question 38: Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using MF with bias-based collaborative filter predictions. Also, plot the average recall (Y-axis) against t (X-axis) and average precision (Y-axis) against average recall (X-axis). Use optimal number of latent factors found in question 25 and sweep t from 1 to 25 in step sizes of 1. For each plot, briefly comment on the shape of the plot.

Table 14: Average Precision and Recall Using MF with Bias Collaborative Filter Predictions

MF: precision vs. t The magnitude of average precisions all remain relatively high (between 0.88 and 0.94). There is a slight downward trend as t increases. Intuitively this makes sense, as it is likely that the model's one highest predicted rating also happens to have a rating above 3.0 in reality. It is more difficult to maintain this consistently across, say 20 predicted ratings, which is why we might see a slight decrease over t.	
MF: recall vs. t As expected, the recall increases with increasing t. It reaches a recall of about 0.7 with the t = 25 (highest t). For t = 1, the recall is 1 if the model guesses the highest predicted movie, and is 0 otherwise. Guessing the one favorite movie is intuitively more difficult than guessing a group of favorite movies, which may explain this increasing trend we observe	

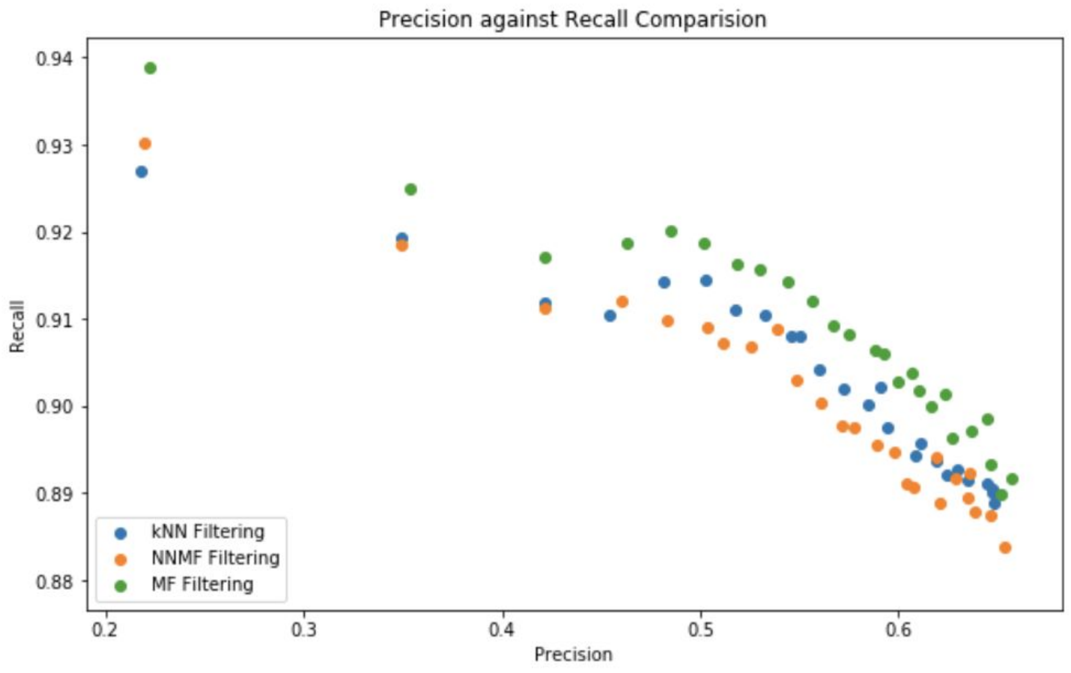
MF: precision vs. recall

As explained earlier, it is intuitive that precision tends to decrease as t increases and that recall tends to increase as t increases. Since these are both shown in the first two plots, it also follows that precision should decrease as a function of increasing recall, as that is associated with increasing t.



Question 39: Plot the precision-recall curve obtained in questions 36, 37, and 38 in the same figure. Use this figure to compare the relevance of the recommendation list generated using k-NN, NNMF, and MF with bias predictions.

Figure 17: Precision-Recall Curves for the k-NN, NNMF, and MF with Bias Collaborative Filters



The results from Figure 17 echo the results in Figure 16, in which the ROC curves were presented. Again, MF with bias filtering appears to have the best recall in general, followed by k-NN collaborative filtering and then NNMF collaborative filtering. These results fit with our previous results in the various collaborative filtering method sections.

Conclusion

This project is a well designed introduction to building a collaborative filtering system. We used different collaborative filtering models such as kNN, NMF, and SVD (biased) to study and compare the performance metrics such as RMSE and MAE. Furthermore, we also studied the precision and recall metrics of the those models. We found that all three models are similar in terms of performance, however, MF (using SVD biased) does perform better than kNN and NMF. We recommend any of the models here is good to begin creating a collaborative filtering system.