ECE219 Large-Scale Data Mining: Models and Algorithms **(Due: Thu, 02/20/20)**

Project 3: Collaborative Filtering

Instructor: Vwani Roychowdhury Zhaoxi Yu(005432230) & Yuhao Yin(104880239)

1 Introduction

The increasing importance of the web as a medium for electronic and business transactions has served as a driving force for the development of recommendation systems technology. An important catalyst in this regard is the ease with which the web enables users to provide feedback about their likes or dislikes. The basic idea of recommendation systems is to utilize these user data to infer customer interests. The entity to which the recommendation is provided is referred to as the user, and the product being recommended is referred to as an item.

The basic models for recommendation systems works with two kinds of data:

1. User-Item interactions such as ratings
2. Attribute information about the users and items such as textual profiles or relevant keywords

Models that use type 1 data are referred to as collaborative filtering methods, whereas models that use type 2 data are referred to as content based methods. In this project, we will build recommendation system using collaborative filtering methods.

2 Collaborative Filtering Models

Collaborative filtering models use the collaborative power of the ratings provided by multiple users to make recommendations. The main challenge in designing collaborative filtering methods is that the underlying ratings matrices are sparse. Consider an example of a movie application in which users specify ratings indicating their like or dislike of movies. Most users would have viewed only a small fraction of the large universe of available movies and as a result, most of the ratings are unspecified.

The basic idea of collaborative filtering methods is that these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items. For example, consider two users named John and Molly, who have very similar tastes. If the ratings, which both have specified, are very similar, then their similarity can be identified by the filtering algorithm. In such cases, it is very likely that the ratings in which only one of them has specified a value, are also similar. *This similarity can be used to make inferences about incompletely specified values.* Most of the collaborative filtering methods focuses on leveraging either inter-item correlations or inter-user correlations for the prediction process.

In this project, we will implement and analyze the performance of two types of collaborative filtering methods:

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

3 MovieLens Dataset

In this project, we will build a recommendation system to predict the ratings of the movies in the MovieLens dataset.

Although the dataset contains movie genre information, we will only use the movie rating information in this project. For the subsequent discussion, we assume that the ratings matrix is denoted by R, and it is an $m \times n$ matrix containing m users (rows) and n movies (columns). The (i, j) entry of the matrix is the rating of user i for movie j and is denoted by r_{ij}. Before moving on to the collaborative filter implementation, we will analyze and visualize some properties of this dataset.

QUESTION 1. Compute the sparsity of the movie rating dataset, where sparsity is defined by equation 1.

$$Sparsity = \frac{Total\ number\ of\ available\ ratings}{Total\ number\ of\ possible\ ratings} \quad (1)$$

Remark. Sparsity of MovieLens Dataset is **0.0170**, which implies that a lot of movie rating information is actually missing.

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.

Remark. Most users give the rating of 4. There is a roughly increase in the ratings of movies, which means the users tend to provide positive comments on movies.

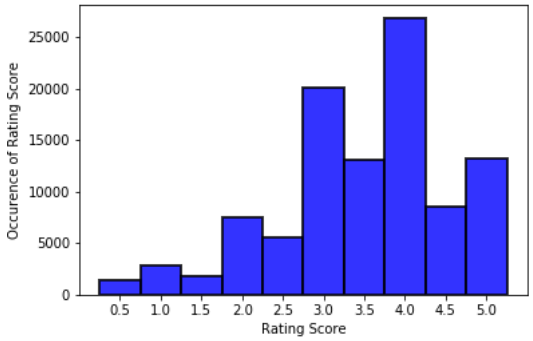

Figure 1: Frequency of Rating Values

QUESTION 3. Plot the distribution of the number of ratings received 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. For example, the movie that has the largest number of ratings has index 1; ties can broken in any way. A monotonically decreasing curve instead of a histogram is expected.

Remark. The largest number of ratings received per movie is more than 300, while around 20% of the movies receive fewer than 10 ratings, which implies difficulty of rating prediction.

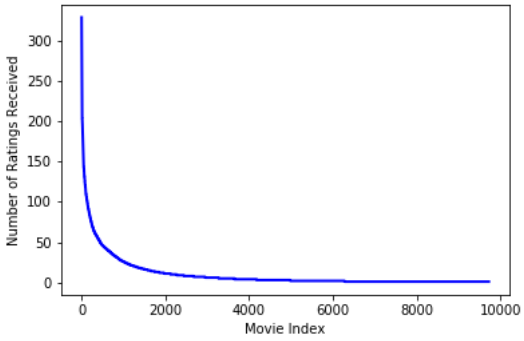

Figure 2: Distribution of Ratings among Movies

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. The requirement of the plot is similar to that in Question 3.

Remark. The largest number of ratings given by one user is more than 2500, while also around 20% of the users seldom rate movies in this dataset.

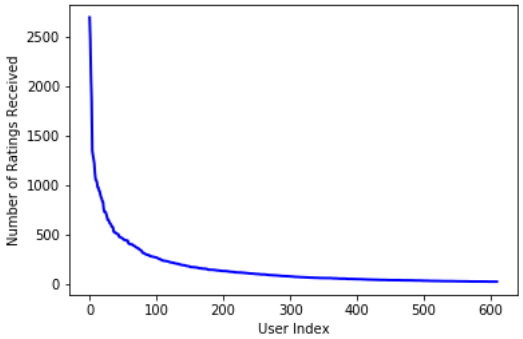

Figure 3: Distribution of Ratings among Users

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

Remark. From the result of Question 3, we observed that only a few movies have considerable quantity of ratings. It is intuitively reasonable that popular movies are often watched by a large number of users, and hence get rated substantially more than unpopular movies. As a result, the recommendation system tends to recommend popular 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.

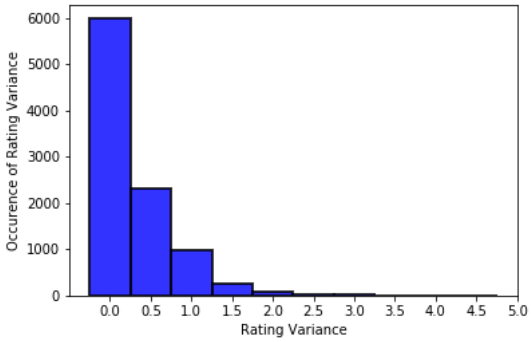

Figure 4: Frequency of Rating Variance

Remark. The variance of rating is in decreasing order, which shows more than half of the movies received very similar ratings, and almost no movies receive ratings that differ more than 2.5. This histogram makes sense, since users tend to have similar judgments on movies, i.e., most of the movies are universally good or bad regardless of the audience.

4 Neighborhood-based Collaborative Filtering

The basic idea in neighborhood-based methods is to use either user-user similarity or item-item similarity to make predictions from a ratings matrix. In this project, we will only implement user-based collaborative filtering (implementation of item-based collaborative filtering is very similar).

4.1 User-based neighborhood models

In this approach, user-based neighborhoods are defined in order to identify similar users to the target user for whom the rating predictions are being computed. In order to determine the neighborhood of the target user u, her similarity to all the other users is computed. Therefore, a similarity function needs to be defined between the ratings specified by users. In this project, we will use Pearson-correlation coefficient to compute the similarity between users.

4.2 Pearson-correlation coefficient

Pearson-correlation coefficient between users u and v, denoted by Pearson(u, v), captures the similarity between the rating vectors of users u and v. Before stating the formula for computing Pearson(u, v), let's first introduce some notation:

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.

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

Remark.

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|} \tag{2}$$

QUESTION 8. In plain words, explain the meaning of $I_u \cap I_v$. Can $I_u \cap I_v = \emptyset$?

Remark. $I_u \cap I_v$ means set of movie indices for which ratings have been specified by both user u and user v. It can be empty set, because movies can be neither rated by user u nor user v.

Then, with the above notation, the Pearson-correlation coefficient between users u and v is defined by equation 3:

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

4.3 k-Nearest neighborhood (k-NN)

Having defined similarity metric between users, now we are ready to define neighborhood of users. k-Nearest neighbor of user u, denoted by P_u, is the set of k users with the highest Pearson-correlation coefficient with user u.

4.4 Prediction function

We can now define the prediction function for user-based neighborhood model. The predicted rating of user u for item j, denoted by $\hat{r}_{uj}$, is given by equation 4.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u} Person(u, v)(r_{vj} - \mu_v)}{\sum_{v \in P_u} | \, Person(u, v) \, |} \tag{4}$$

QUESTION 9. Can you explain the reason behind mean-centering the raw ratings $(r_{vj} - \mu_v)$ in

the prediction function?

Remark. Mean-centering will reduce the impact of different rating habits of users. For instance, user u and user v have similar taste for movies. However, user u tends to rate more strictly than user v. Without mean-centering, the recommendation system may give inaccurate prediction. By applying equation 4, ratings are balanced and potential biases are removed.

4.5 k-NN collaborative filter

The previous sections have equipped you with the basics needed to implement a k-NN collaborative filter for predicting ratings of the movies. Although, we have provided you with the equations needed to write a function for predicting the ratings but we don't require you to write it. Instead, you can use the built-in python functions for prediction.

4.5.1 Design and test via cross-validation

In this part of the project, you will design a k-NN collaborative filter and test it's performance via 10-fold cross validation. In a 10-fold cross-validation, the dataset is partitioned into 10 equal sized subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the filter, and the remaining 9 subsets are used to train the filter. The cross-validation process is then repeated 10 times, with each of the 10-subsets used exactly once as the validation data.

QUESTION 10. Design a k-NN collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate it's 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).

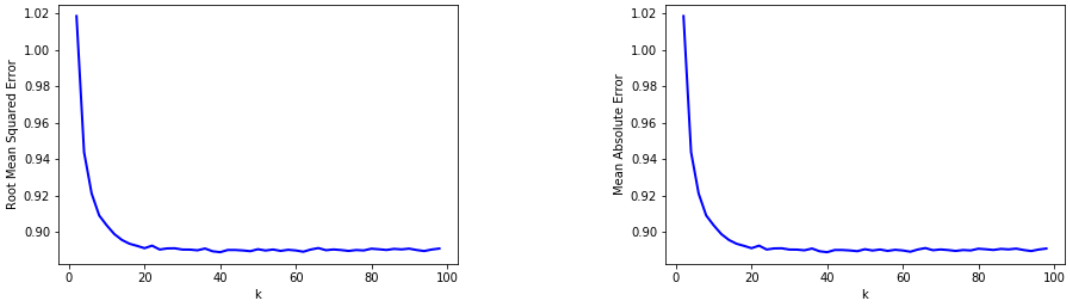

Figure 5: k-NN-based Collaborative Filter Performance Evaluation through Cross-Validation

Remark. Based on the cross-validation result, we observe a nearly-monotonic decreasing trend of both RMSE and MAE when the number of neighbors k varies from 2 to 100, indicating prediction based on more similar users always incurs better performance.

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.

Remark. Minimum k for RMSE criterion is **24**, corresponding minimal average RMSE is **0.8903**. Minimum k for MAE criterion is **18**, corresponding minimal average MAE is **0.6801**.

4.6 Filter performance on trimmed test set

In this part of the project, we will analyze the performance of the k-NN collaborative filter in predicting the ratings of the movies in the trimmed test set. The test set can be trimmed in many ways, but we will consider the following trimming:

- **Popular movie trimming**: In this trimming, we trim the test set to contain movies that have received more than 2 ratings. To be specific, if a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
- **Unpopular movie trimming**: In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings. To be specific, if a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
- **High variance movie trimming**: In this trimming, we trim the test set to contain movies that have variance (of the rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. To be specific, if a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

QUESTION 12. Design a k-NN collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it's 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.

Remark. The minimum average RMSE after popular movie trimming is **0.8725**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 6,

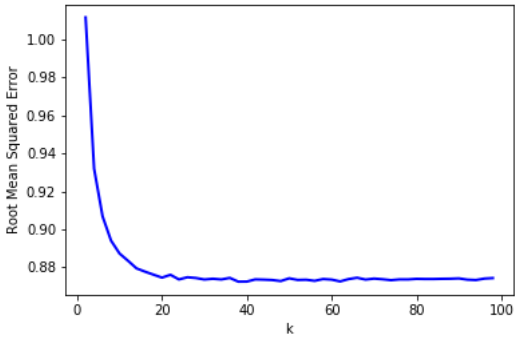

Figure 6: k-NN Collaborative Filter Performance after Popular Movie Trimming

Remark 2. RMSE for popular movie trimmed test set is very close to that without trimming operation, indicating that removing unpopular movies from testing dataset actually stabilized the testing performance, since those unpopular movies have few ratings and thus are hard to predict the ratings for major users.

QUESTION 13. Design a k-NN collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate it's 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.

Remark. The minimum average RMSE after unpopular movie trimming is **1.1112**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 7,

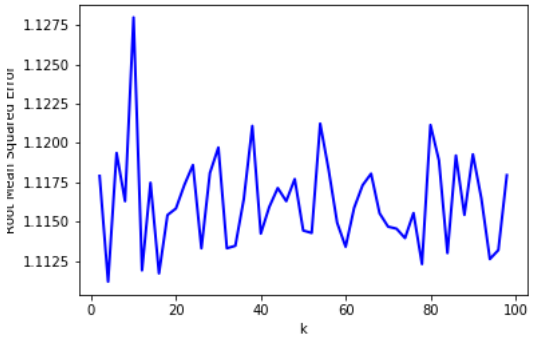

Figure 7: k-NN Collaborative Filter Performance after Unpopular Movie Trimming

Remark 2. We would argue the validity of k-NN Collaborative Filter model for unpopular movie trimmed test set, since all movies left in the test dataset after the trimming operation receive few ratings by users. Consequently, finding neighbors in the training dataset is difficult, sometimes impossible, since the corresponding column of rating matrix is very sparse.

Remark 3. Frankly speaking, the jumpy pattern in Figure 7 is actually caused by different data splitting scheme rendered by different random states in the cross-validation, instead of actually capturing the relationship between RMSE and different choices of k. Moreover, the difference between RMSE is so minor, and if we split the whole dataset always in the same way, then the corresponding curve is indeed a horizontal line.

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 it's 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.

Remark. The minimum average RMSE after high variance movie trimming is **1.4388**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 8,

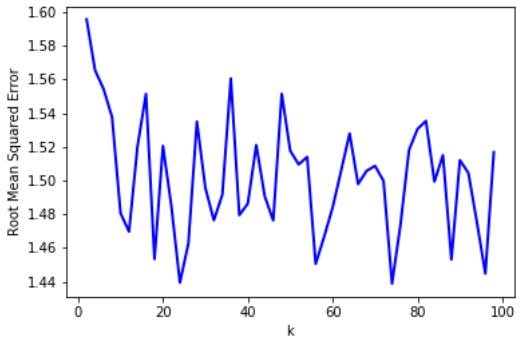

Figure 8: k-NN Collaborative Filter Performance after High Variance Movie Trimming

Remark 2. Same situation in the unpopular movie trimmed test set applies here. According to Question 6, we know that few movies possess high variance of ratings. As a result, predicting on test dataset after high variance movie trimming will cause the same problem as before: there won't be enough number of neighbors in the training dataset, and simply predicting with the global mean is the only choice. This problem becomes more severe especially when k grows relatively large, around 20-30 based on our experiments.

4.6.1 Performance evaluation using ROC curve

Receiver operating characteristic (ROC) curve is a commonly used graphical tool for visualizing the performance of a binary classifier. It plots the true positive rate (TPR) against the false positive rate (FPR).

In the context of recommendation systems, it is a measure of the relevance of the items recommended to the user. Since the observed ratings are in a continuous scale (0-5), so we first need to convert the observed ratings to a binary scale. This can be done by thresholding the observed ratings. If the observed rating is greater than the threshold value, then we set it to 1 (implies that the user liked the item). If the observed rating is less than the threshold value, then we set it to 0 (implies that the user disliked the item). After having performed this conversion, we can plot the ROC curve for the recommendation system in a manner analogous to that of a binary classifier.

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.

Remark. The optimal number of neighbors found in Question 11 yields $k = 24$ using RMSE criterion. ROC curves and respective AUC scores using different threshold values are plotted in Figure 9

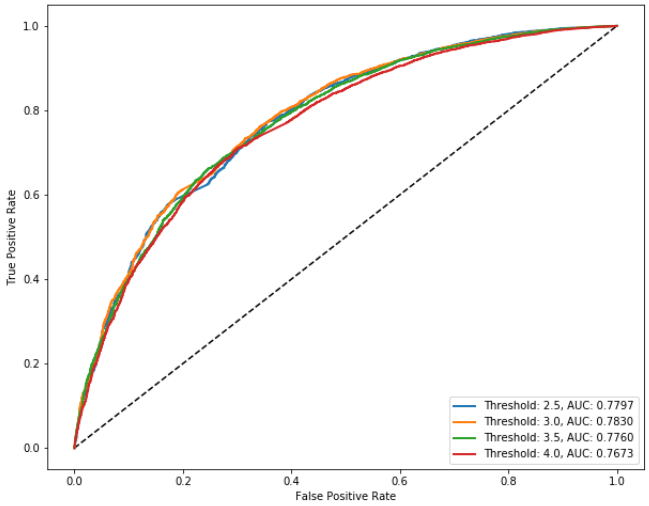

Figure 9: ROC Curves for k-NN-based Collaborative Filter

5 Model-based Collaborative Filtering

In model-based collaborative filtering, models are developed using machine learning algorithms to predict users' rating of unrated items. Some examples of model-based methods include decision trees, rule-based models, bayesian methods, and latent factor models. In this project, we will explore *latent factor based models* for collaborative filtering.

5.1 Latent Factor based Collaborative Filtering

Latent factor based models can be considered as a direct method for matrix completion. It estimates the missing entries of the rating matrix R, to predict what items a user will most probably like other than the ones they have rated. The basic idea is to exploit the fact that *significant portions of the rows and columns of the rating matrix are correlated.* As a result, the data has built-in redundancies and the rating matrix R can be approximated by a low-rank matrix. The low-rank matrix provides a robust estimation of the missing entries.

The method of approximating a matrix by a low-rank matrix is called matrix factorization. The matrix factorization problem in latent factor based model can be formulated as an optimization problem given by

$$\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{i=1}^{n}(r_{ij} - (UV^\top)_{ij})^2 \tag{5}$$

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

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

Then, we can reformulate the optimization problem as

$$\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{i=1}^{n} W_{ij}(r_{ij} - (UV^\top)_{ij})^2 \tag{6}$$

Since the rating matrix R is sparse, the observed set of ratings is very small. As a result, it might cause over-fitting. A common approach to address this problem is to use regularization. The regularization parameter λ in the following formulation is always non-negative and it controls the weight of the regularization term:

$$\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{i=1}^{n} W_{ij}(r_{ij} - (UV^\top)_{ij})^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 \tag{7}$$

There are many variations to the unconstrained matrix factorization formulation depending on the modification to the objective function and the constraint set. In this project, we will explore two such variations:

- **Non-negative matrix factorization (NNMF)**
- **Matrix factorization with bias (MF with bias)**

5.2 Non-Negative Matrix Factorization (NNMF)

Non-negative matrix factorization may be used for ratings matrices that are non-negative. As we have seen in the lecture, that the major advantage of this method is the high level of interpretability it provides in understanding the user-item interactions. The main difference from other forms of matrix factorization is that the latent factors U and V must be non-negative. Therefore, optimization formulation in non-negative matrix factorization is given by,

$$\begin{aligned} &\underset{U,V}{\text{minimize}} \sum_{i=1}^{m}\sum_{i=1}^{n} W_{ij}(r_{ij} - (UV^\top)_{ij})^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 \\ &\text{subject to } \; U \geq 0, V \geq 0 \end{aligned} \tag{8}$$

There are many optimization algorithms like stochastic gradient descent (SGD), alternating least-squares (ALS), etc for solving this optimization problem. Since you are familiar with the SGD method, we will not describe it here. Instead, we will provide the motivation and main idea behind the ALS algorithm. SGD is very sensitive to initialization and step size. ALS is less sensitive to initialization and step size, and therefore a more stable algorithm than SGD. ALS also has a faster convergence rate than SGD. The main idea in ALS, is to keep U fixed and then solve for V. In the next stage, keep V fixed and solve for U. In this algorithm, at each stage we are solving a least-squares problem.

Remark. Although ALS has a faster convergence rate and is more stable, we will use SGD in this project. The main reason behind this is based on the fact that the python package that we will be using to design the NNMF-based collaborative filter only has the SGD implementation. This choice would have no effect on the performance of the filter designed because both the SGD and ALS converges for the MovieLens dataset. The only downside of using SGD is that it will take a little bit longer to converge, but that will not be a big issue as you will see while designing the NNMF filter.

QUESTION 16. Is the optimization problem given by equation 6 convex? If not, consider the optimization problem given by equation 6. For U fixed, formulate it as a least-squares problem.

Proof. To check the convexity of this optimization problem, we are going to prove its corresponding Hessian matrix is not positive semi-definite. For simplicity, assume $m = n = k = 1$ and $W_{11} = 1$. Then, the optimization problem can be reformulated as,

$$L(U, V) = \frac{1}{2}(R - UV)^2$$

Note it makes no difference if we multiple the objective function by some constant. Calculate the Hessian matrix of L w.r.t. U, V:

$$\nabla^2 L(U, V) = \begin{bmatrix} \frac{\partial^2 L}{\partial U^2} & \frac{\partial^2 L}{\partial U \partial V} \\ \frac{\partial^2 L}{\partial V \partial U} & \frac{\partial^2 L}{\partial V^2} \end{bmatrix} = \begin{bmatrix} V^2 & -R + 2UV \\ -R + 2UV & U^2 \end{bmatrix}$$

The determinant of the Hessian matrix follows,

$$|\nabla^2 L(U, V)| = -(R - UV)(R - 3UV)$$

which is not positive semi-definite. Therefore, for general m, n, k, the objective function is non-convex as well.

For fixed U, the objective function becomes,

$$L(V) = \frac{1}{2}\sum_{j=1}^{n}(r_j - UV_j)^\top W_j(r_j - UV_j)$$

where $r_j = (r_{1j}, \cdots, r_{mj})^\top$, $V_j = (V_{1j}, \cdots, V_{kj})^\top$, and $W_j = \text{diag}(W_{1j}, \cdots, W_{mj})$. This is the formulation of a weighted least-squares problem, and the corresponding optimal V is given by,

$$V_j^* = (U^\top W_j U)^{-1} U^\top W_j r_j, \quad j = 1, \cdots, n.$$

Hence, for each stage of ALS algorithm, we are essentially solving a least-squares problem, which makes the algorithm stable and possessing a faster convergence rate. □

5.2.1 Prediction Function

After we have solved the optimization problem in equation 8 for U and V, then we can use them for predicting the ratings.The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$, is given by

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

Having covered the basics of matrix factorization, now we are ready to implement a NNMF based collaborative filter to predict the ratings of the movies.

5.2.2 Design and Test via Cross-Validation

In this part, we will design a NNMF-based collaborative filter and test its performance via 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 it's 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.

Remark. To implement this NNMF-based collaborative filter, we utilize `matrix_factorization.NMF` from `surprise.prediction_algorithms` package, and set the regularization parameter to be the default 0.06. Afterwards, we calculate the average RMSE and MAE using cross-validation for various k's and report the results as follows,

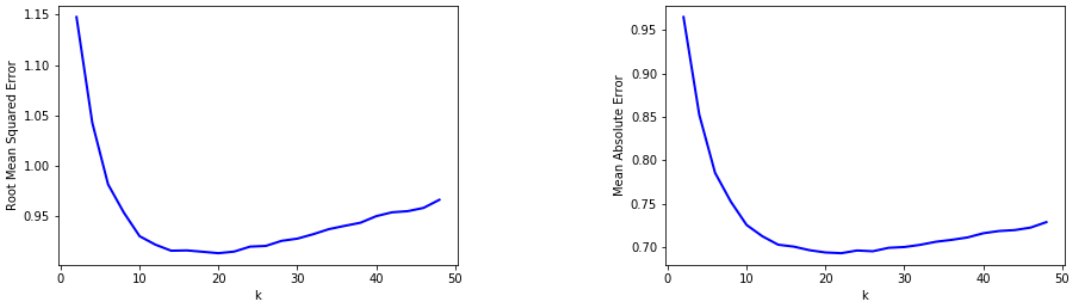

Figure 10: NNMF-based Collaborative Filter Performance Evaluation through Cross-Validation

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?

Remark. The optimal number of latent factors is the value of k that minimizes the average RMSE or MAE. Results from NNMF-based collaborative filter are reported in the following table,

Optimal Latent Factors (minimum RMSE)	**Optimal Latent Factors (minimum MAE)**	**Movie Genres**
20 (0.9136)	22 (0.6930)	19

Table 1: Comparison between optimal number of latent factors and movie genres

Remark 2. The optimal number of latent factors using either RMSE or MAE criterion, is very close to the number of movie genres, which provides a reasonable explanation or interpretation of latent factors.

5.2.3 NNMF Filter Performance on Trimmed Test Set

Having designed the NNMF filter in the previous section, now we will test the performance of the filter in predicting the ratings of the movies in the trimmed test set.

QUESTION 19. Design a NNMF collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it's 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.

Remark. The minimum average RMSE after popular movie trimming is **0.8932**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 11,

Remark 2. Pattern of the plot after popular movie trimming is very close to that without trimming operation, indicating that removing unpopular movies from testing dataset actually stabilized the testing performance, since those unpopular movies have few ratings and thus are hard to predict the ratings for major users.

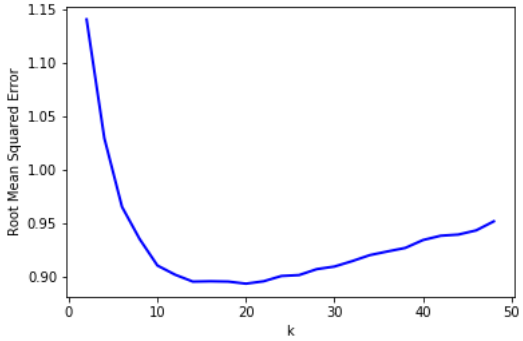

Figure 11: Average RMSE after Popular Movie Trimming for NNMF-based Collaborative Filter

QUESTION 20. Design a NNMF collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate it's 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.

Remark. The minimum average RMSE after unpopular movie trimming is **1.1738**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 12,

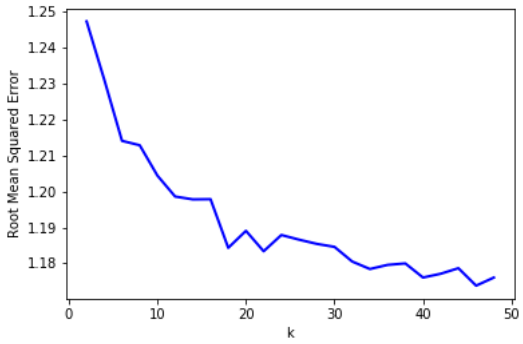

Figure 12: Average RMSE after Unpopular Movie Trimming for NNMF-based Collaborative Filter

Remark 2. For unpopular movie trimming, we observe some nearly-monotonically decreasing pattern in the average RMSE plot. That's probably due to the reason that for sparse data, we need more latent factors to retain numerical information.

QUESTION 21. Design a NNMF collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate it's 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.

Remark. The minimum average RMSE after high variance movie trimming is **1.5311**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 13,

Remark 2. After high variance movie trimming operation, types of movies left in the testing dataset are mainly controversial. Here, being controversial means the movie is somewhat popular (having relatively large number of ratings), but ratings provided by different users vary a lot (having large variance). Hence, prediction on those movies is pretty hard, since *to a thousand readers, there are a thousand Hamlets.* That's why we have those jumpy patterns in the plot.

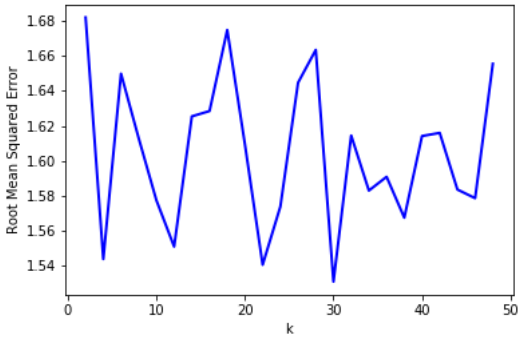

Figure 13: Average RMSE after High Variance Movie Trimming for NNMF-based Collaborative Filter

5.2.4 Performance Evaluation using ROC Curve

In this part, we will evaluate the performance of the NNMF-based collaborative 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 plotting 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.

Remark. The optimal number of latent factors found in QUESTION 18 yields $k = 20$ using RMSE criterion. ROC curves and respective AUC scores using different threshold values are summarized in Figure 14,

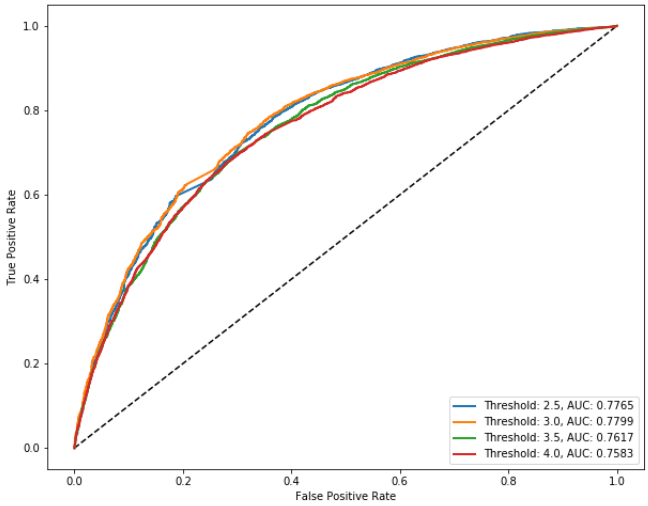

Figure 14: ROC Curves for NNMF-based Collaborative Filter with Different Thresholds

Remark 2. Threshold 3 renders highest AUC scores, which is **0.7799**.

5.2.5 Interpretability of NNMF

The major advantage of NNMF over other forms of matrix factorization is not necessarily one of accuracy, but that of the high level of interpretability it provides in understanding user-item interactions. In this part of the project, we will explore the interpretability of NNMF. Specifically, we will explore the connection between the latent factors and the movie genres.

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?

Remark. For this question, we are trying to explore whether there exists a connection between latent factors and movie genres. For concise illustration, we report three representative latent factors in Table 2, and provide the corresponding analysis.

Latent Factor 1	Latent Factor 13	Latent Factor 14
Drama	Action|Thriller	Comedy|Romance
Horror|Thriller	Comedy|Drama	Action|Adventure|Fantasy
Children|Comedy	Documentary	Comedy|Crime|Romance|Thriller
Comedy|Drama|Romance	Drama|Romance|War	Comedy|Romance
Drama	Drama|Horror|Thriller	Comedy|Romance
Children|Comedy	Drama	Sci-Fi
Drama	Comedy|Drama|Romance	Drama|Film-Noir|Romance
Crime|Drama	Crime|Drama|Thriller	Comedy|Drama
Comedy|Crime	Comedy|Drama	Comedy|Drama
Crime|Drama	Horror|Thriller	Thriller

Table 2: Genres of Top 10 Movies for Three Representative Latent Factors

Remark 2. For latent factor 1, most of the top 10 movies are dramas and crimes, indicating latent factor 1 represents some combination of drama and crime movies. Similarly, latent factor 13 captures drama, horror and thriller movies, while latent factor 14 is strongly tied to comedy and romance movies.

5.3 Matrix Factorization with Bias (MF with Bias)

In MF with bias, we modify the cost function (equation 7) by adding bias term for each user and item. With this modification, the optimization formulation of MF with bias is given by

$$\underset{U,V,b_u,b_i}{\text{minimize}} \sum_{i=1}^{m}\sum_{i=1}^{n} W_{ij}(r_{ij} - (UV^\top)_{ij})^2 + \lambda\|U\|_F^2 + \lambda\|V\|_F^2 + \lambda\sum_{u=1}^{m} b_u^2 + \lambda\sum_{i=1}^{n} b_i^2 \tag{10}$$

In the above formulation, b_u is the bias of user u and b_i is the bias of item i, and we jointly optimize over U, V, b_u, b_i to find the optimal values.

5.3.1 Prediction Function

After we have solved the optimization problem in equation 10 for U, V, b_u, b_i, then we can use them for predicting the ratings. The predicted rating of user i for itme j, denoted by $\hat{r}_{ij}$ is given by

$$\hat{r}_{ij} = \mu + b_i + b_j + \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{11}$$

where μ is the mean of all ratings, b_i is the bias of user i, and b_j is the bias of item j.

5.3.2 Design and Test via Cross-Validation

In this part, you will design a MF with bias collaborative filter and test it's performance via 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 it's 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.

Remark. To implement this MF with bias collaborative filter, we utilize `matrix_factorization.SVD` from `surprise.prediction_algorithms` package, and set the regularization parameter to be the default 0.06. Afterwards, we calculate the average RMSE and MAE using cross-validation for various k's and report the results as follows,

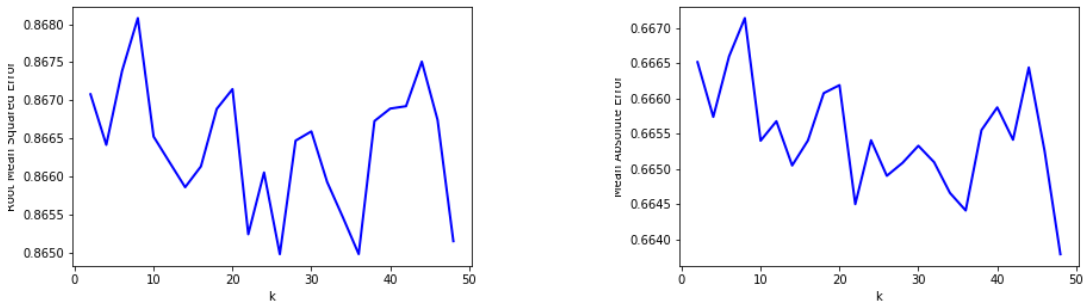

Figure 15: MF with Bias Collaborative Filter Performance Evaluation through Cross-Validation

Remark 2. Surprisingly, matrix factorization with bias collaborative filter does not yield a consistent performance, although overall there is a decreasing trend as k increases for both average RMSE and average MAE. However, differences between those values are so narrow that we highly recommend to expand the searching range of k in order to figure out the true relationship.

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.

Remark. The optimal number of latent factors is the value of k that minimizes the average RMSE or MAE. Results from MF with bias collaborative filter are reported in the following table,

Optimal Latent Factors (minimum RMSE)	**Optimal Latent Factors (minimum MAE)**	**Movie Genres**
26 (0.8650)	48 (0.6638)	19

Table 3: Comparison between optimal number of latent factors and movie genres

Remark 2. The optimal number of latent factors from both average RMSE and average MAE criteria are no close to the number of movie genres for MF with bias collaborative filter. Therefore, latent factors in this model are not very interpretable. Also, model performances across different number of latent factors are nearly identical. Therefore, if we re-run cross-validation with a different random seed, it's highly possible that the corresponding optimal number of latent factors will be entirely different.

5.3.3 MF with Bias Filter Performance on Trimmed Test Set

Having designed the MF with bias filter in the previous section, now we will test the performance of the filter in predicting the ratings of the movies in the 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 it's 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.

Remark. The minimum average RMSE after popular movie trimming is **0.8576**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 16,

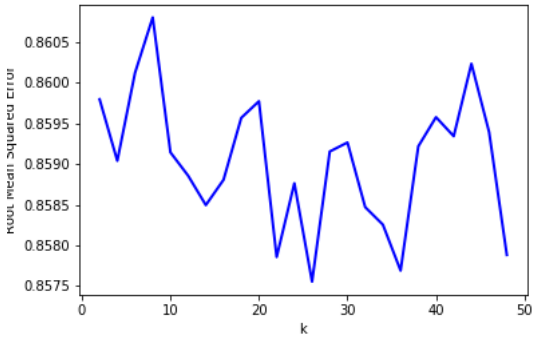

Figure 16: Average RMSE after Popular Movie Trimming for MF with Bias Collaborative Filter

Remark 2. Similarly, plot of average RMSE after popular movie trimming is nearly identical to that without trimming. Justifications concerning this phenomenon have already been discussed before.

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 it's 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.

Remark. The minimum average RMSE after unpopular movie trimming is **0.9715**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 17,

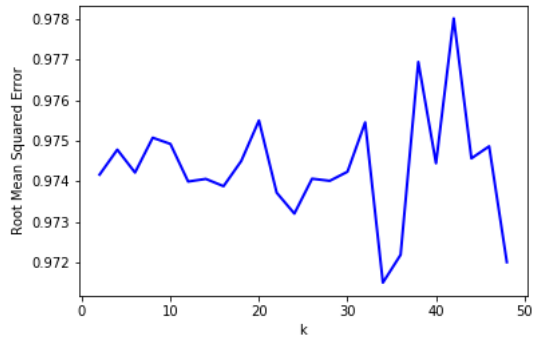

Figure 17: Average RMSE after Unpopular Movie Trimming for MF with Bias Collaborative Filter

Remark 2. For MF with bias collaborative filter with unpopular movie trimmed test set, there isn't much fluctuation in terms of average RMSE, since the maximum and minimum average RMSE differ by 0.0065 for the number of latent factors ranging from 2 to 50.

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 it's 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.

Remark. The minimum average RMSE for high variance movie trimmed test set is **1.4377**. Besides, plot showing the trend of average RMSE over different values of k is reported accordingly in Figure 18,

Remark 2. Average RMSE for high variance movie trimmed test set is much higher than those for popular or unpopular movie trimmed ones. As we have already explained before, predicting ratings given by new users for those

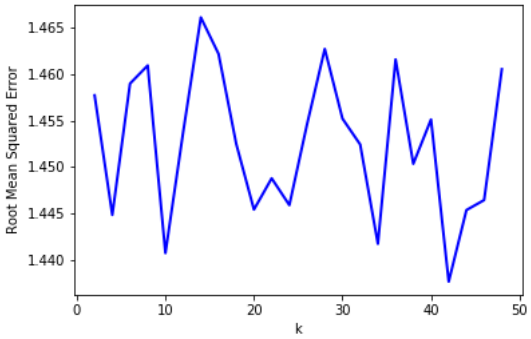

Figure 18: Average RMSE after High Variance Movie Trimming for MF with Bias Collaborative Filter

5.3.4 Performance Evaluation using ROC Curve

In this part, we will evaluate the performance of the MF with bias collaborative 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.

Remark. The optimal number of latent factors found in QUESTION 25 yields $k = 26$ using RMSE criterion. ROC curves and respective AUC scores using different threshold values are summarized in Figure 19,

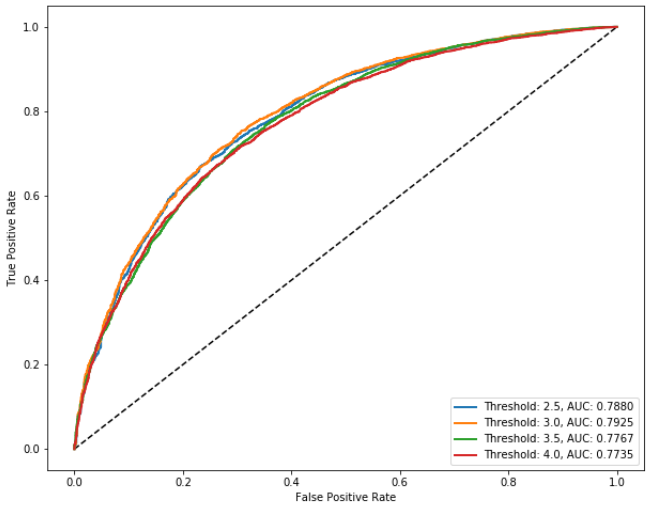

Figure 19: ROC Curves for MF with Bias Collaborative Filter with Different Thresholds

Remark 2. Same as the conclusion from NNMF-based Collaborative Filter, threshold 3 renders the highest AUC score, which is **0.7925**.

6 Naive Collaborative Filtering

In this part of the project, we will implement a naive collaborative filter to predict the ratings of the movies in the MovieLens dataset. This filter returns the mean rating of the user as it's predicted rating for an item.

6.1 Prediction Function

The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$ is given by,

$$\hat{r}_{ij} = \mu_i \tag{12}$$

where μ_i is the mean rating of user i.

6.2 Design and Test via Cross-Validation

Having defined the prediction function of the naive collaborative filter, we will design a naive collaborative filter and test it's performance via 10-fold cross validation.

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

Remark. For naive collaborative filter, we simply predict the average ratings among all training data given any new user. As we can imagine, this naive model should render higher RMSE compared to both neighborhood-based and model-based collaborative filters. After defining this new prediction algorithm using `surprise.AlgoBase`, we calculate the average RMSE to be **1.0425**.

Remark 2. An important thing to note about the naive collaborative filter is that there is no notion of training. For solving QUESTION 30, split the dataset into 10 pairs of train set and test set and for each pair predict the ratings of the movies in the test set using the prediction function (no model fitting required). Then compute the RMSE for this fold and repeat the procedure for all the 10 folds. The average RMSE is computed by averaging the RMSE across all the 10 folds.

6.3 Naive Collaborative Filter Performance on Trimmed Test Set

Having designed the naive collaborative filter in the previous section, now we will test the performance of the filter in predicting the ratings of the movies in the 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 it's performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

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

QUESTION 33. Design a naive collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate it's perfor- mance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

Remark. We shall answer aforementioned three questions together in table 4, since they are very similar to each other and there is only one value, i.e., average RMSE, to be reported for each trimmed test set.

Popular Movie	1.0357
Unpopular Movie	1.1438
High Variance Movie	1.6385

Table 4: Naive Collaborative Filter Performances on Different Trimmed Test Sets

Remark 2. Not unexpectedly, naive collaborative filter on high variance movie trimmed test set yields the worst performance, and performs best on popular movie trimmed test set.

7 Performance Comparison

In this section, we will compare the performance of the various collaborative filters (designed in the earlier sections) in predicting the ratings of the movies in the MovieLens dataset.

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.

Remark. In order to compare model performances in terms of predicting movie ratings, we plot three ROC curves in the same figure as shown in Figure 20 for respective collaborative filters using the same threshold 3. Since for all collaborative filtering models, threshold being equal to 3 always renders the highest AUC score, this is a fair comparison.

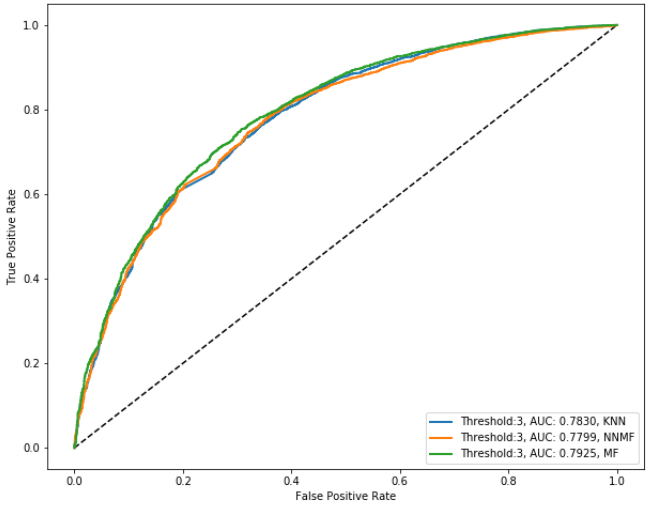

Figure 20: ROC Curves for Performance Comparisons among Three Collaborative Filters

Remark 2. Among these three different collaborative filters, MF with bias-based collaborative filter achieves the highest AUC score, which implies that it is the optimal model for this movie rating prediction problem. This makes sense, since compared to NNMF-based collaborative filter, MF with bias-based collaborative filter is more flexible by introducing optimization variables for both user and movie bias. Therefore, for those users who always tend to give a unreasonably high or low rating, MF with bias-based collaborative filter would take that into consideration and try to eliminate the user bias when predicting rating for other users.

Remark 3. On the other hand, NNMF-based collaborative filter performs worst in terms of movie rating prediction. This can be understood as a trade-off between prediction accuracy and model interpretability.

8 Ranking

Two primary ways in which a recommendation problem may be formulated are:

1. *Prediction version of problem:* Predict the rating value for a user-item combination
2. *Ranking version of problem:* Recommend the top k items for a particular user

In previous parts of the project, we have explored collaborative filtering techniques for solving the prediction version of the problem. In this part, we will explore techniques for solving the ranking version of the problem. There are two approaches to solve the ranking problem:

- Design algorithms for solving the ranking problem directly
- Solve the prediction problem and then rank the predictions

Since we have already solved the prediction problem, so for continuity we will take the second approach to solving the ranking problem.

8.1 Ranking Predictions

The main idea of the second approach is that it is possible to rank all the items using the predicted ratings. The ranking can be done in the following manner:

- For each user, compute it's predicted ratings for all the items using one of the collaborative filtering techniques. Store the predicted ratings as a list L.
- Sort the list in descending order, the item with the highest predicted ratings appears first and the item with the lowest predicted ratings appears last.
- Select the first t-items from the sorted list to recommend to the user.

8.2 Evaluating Ranking using Precision-Recall Curve

Precision-recall curve can be used to evaluate the relevance of the ranked list. Before stating the expressions for precision and recall in the context of ranking, let's introduce some notation:

- $S(t)$: The set of items of size t recommended to the user. In this recommended set, ignore (drop) the items for which we don't have a ground truth rating.
- G: The set of items liked by the user (ground-truth positives).

Then with the above notation, the expressions for precision and recall are given by,

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|}, \qquad Recall(t) = \frac{|S(t) \cap G|}{|G|} \tag{13}$$

QUESTION 35. Precision and Recall are defined by the mathematical expressions given by equation 13. Please explain the meaning of precision and recall in your own words.

Remark. Precision measures the percentage of correct recommended items among all the recommendations. It informs us the reliability of predictions rendered by some model.

Remark 2. Recall measures the percentage of correct recommended items among the set of items liked by the user. This gives us a sense of how much of the real liked items (ground-truth positives) got retrieved by the model.

Both precision and recall are functions of the size of the recommended list (t). Therefore, we can generate a precision-recall plot by varying t.

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.

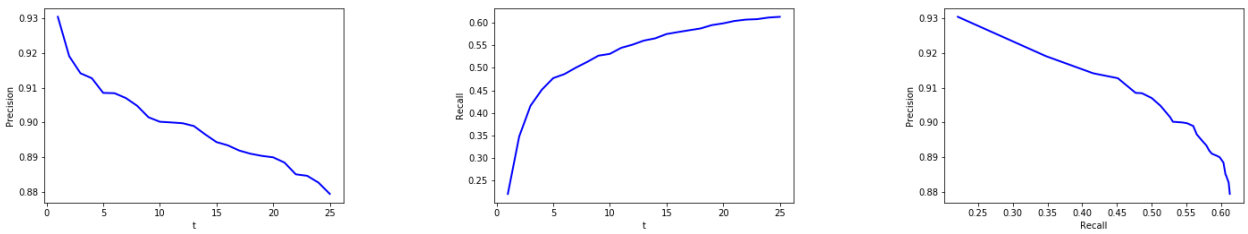

Figure 21: Precision-Recall Curve for KNN Collaborative Filter

Remark. For the first subplot in Figure 21, we observe a decreasing trend of precision when t increases. This makes intuitive sense since if the recommendation system is asked to recommend one single movie

to the user, it will probably hit the ground-truth positives by picking the movie with highest predicted rating. As the number of recommended movies increases, say t becomes 20, then the top 20th movie may not the one liked by the user with some predicted rating relatively closer to the threshold 3.

Remark 2. For the second subplot in Figure 21, there exists a significant improvement in recall especially when t grows in the beginning stage. This can be easily explained by looking at the mathematical expression of the recall: since the denominator, $|G|$, represents the number of movies liked by the user, which is a constant, then when t increases, the intersection between $S(t)$ and G keeps growing, leading to the monotonic increase in recall.

Remark 3. For the last one in Figure 21, as we can expect, there is a negative relationship between precision and recall. Reasons are straightforward: the relationship between precision and t is negative, and the relationship between recall and t is positive. Hence, serving as a latent variable in between, t bridges precision and recall and renders the curve in the last subplot.

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.

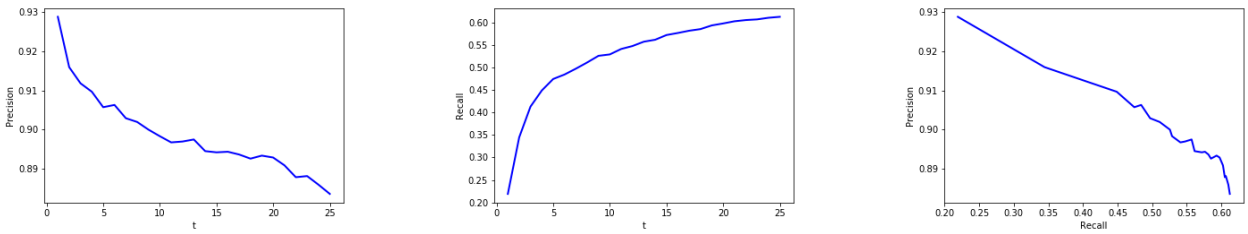

Figure 22: Precision-Recall Curve for NNMF-based Collaborative Filter

Remark. For each plot in Figure 22, the shape of the plot is very similar to the one in Question 36, respective reasoning is similar as well.

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.

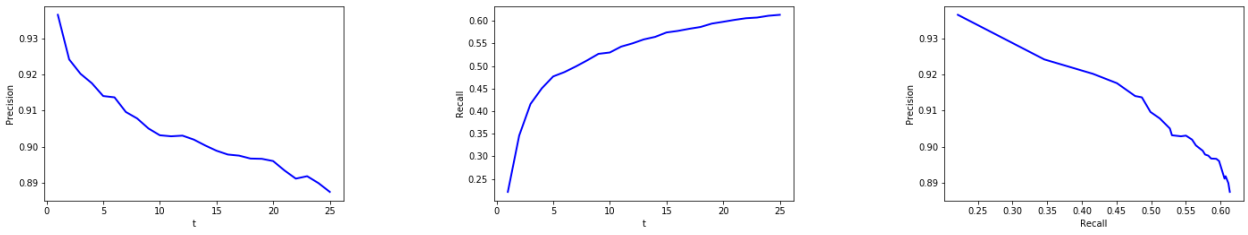

Figure 23: Precision-Recall Curve for MF with bias-based Collaborative Filter

Remark. For each plot in Figure 23, the shape of the plot is very similar to the one in Question 36, respective reasoning is similar as well.

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.

Remark. For precision and recall, we want both of them to be as close to 1 as possible, since they are different measures of prediction accuracy. If we want to use the precision-recall curve to compare the relevance of the recommendation list generated by different collaborative filters, we can simply fix precision (or recall), and compare the recall (or precision) for different models under the same value of precision (or recall). Similar as AUC score for the ROC curve, area under the precision-recall curve can be served as a scalar measure for comparing different recommendation systems' performances. The precision-recall curves are illustrated as follows,

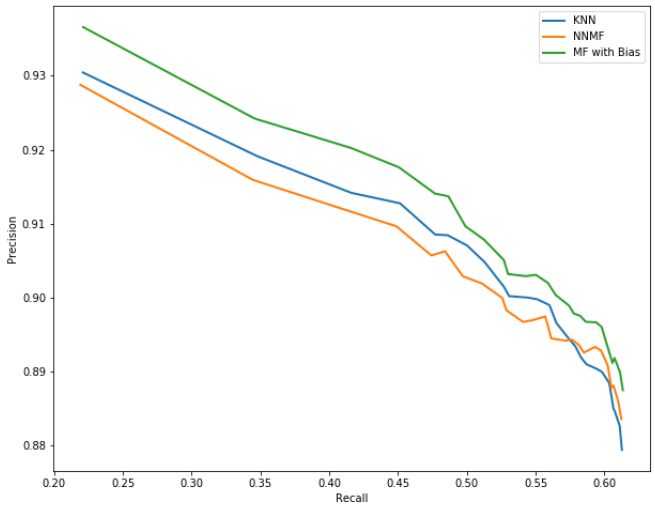

Figure 24: Precision-Recall Curves for Performance Comparisons among Three Collaborative Filters

Remark 2. According to the precision-recall curves in Figure 24, MF with bias-based collaborative filter achieves highest prediction accuracy among all three recommendation systems, while NNMF-based collaborative filter performs worst. The conclusion is identical to that from ROC curves. Similar explanations concerning this result can apply here as well.