**ECE 219 Project 3**

Wei Zhou 505650843 Da Yin 005453741 Amir Dhillon 005649833

### **Introduction**:

In this project, we will explore a few collaborative filtering recommendation models. The dataset we are using is Movielens-small [1]. After preliminary data exploration, we can see below:

There are 100836 ratings total

There are 610 unique users

There are 9724 unique movies.

Ratings are explicit and range from 0.5 to 5

### **QUESTION 1:**

Sparsity is a measure of how separated the dataset is. There are total of 100836 existing ratings, and total number of possible ratings is 610 \* 9724 = 5931640.

By definition from other places,

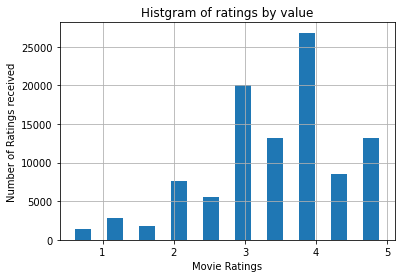
Density of matrix = (available ratings)/(total of possible ratings)=0.0170

Sparsity of matrix = (1- density)= 0.9830

but this class seems to have a unique definition of sparsity, so we are going to use sparsity = 0.0170.

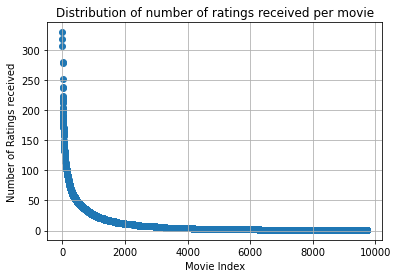
By either formula, we can see that the ratings dataset is very sparse.

### **QUESTION 2:**

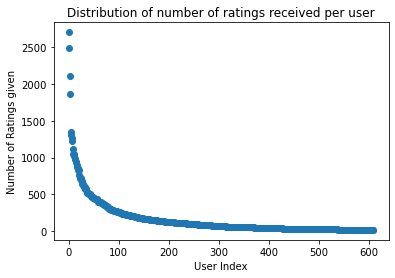


There are many users rating a movie for 3 and 4, and very few for rating 2 and below. Intuitively, we know that users generally give a lot of movies “middle” ratings, and not many high or low ratings.

### **QUESTION 3:**



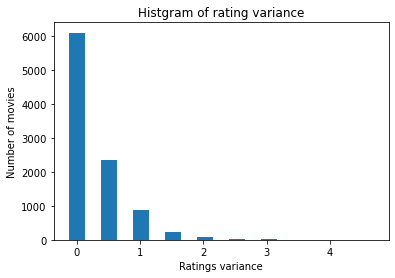
### **QUESTION 4:**



### **QUESTION 5:**

Most of the movies only received a few ratings (less than 10), and there are some popular movies that received a few thousands of ratings. It implies that these movies have a variety amount of ratings, this unbalanced data may create some bias in the learning process. It should require some heavy regularization in order to prevent overfitting. In practice, prediction would be difficult to include movies with low amount of ratings; it would be difficult for users who have not reviewed many movies.

### **QUESTION 6:**



The number of movies for the same variance of ratings decreases as the variance increases. Most movies (about 80%) have a rating variance less than 1, which means their ratings from different users are very similar. However, there are some movies that have high rating variance, which means their ratings are controversial and vary hugely by different users. Intuitively, these high variance movies are good candidates for learning systems because we can use them to cluster users since users' tastes about these movies are quite different.

### **QUESTION 7:**

The mean rating for user u equals the sum of all the ratings this user rated, divided by the total number of movies this user rated.

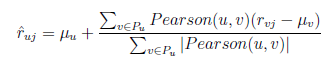


### **QUESTION 8:**

Iu ∩ Iv is the common set of indices of movies that both user u and v have rated.

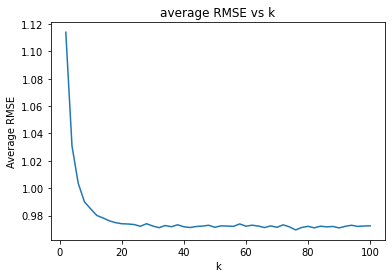
It can be empty because there can exist two users who don't rate any common movies.

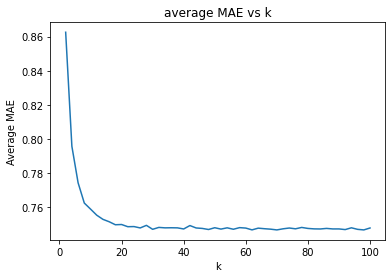
### **QUESTION 9:**



The prediction function is using the average rating of a movie rated by all the similar users v, but directly using the raw rating average will take into account some users rate all movies high, or low. For example, if a user averagely rates the moves 4.5, and rates a movie j 4.0, it means this user does not like this movie that much. Now if we use the mean-centered ratings, the rating contribution this user put for this movie j will be 4.0- 4.5 = -0.5, so it is more realistically reflected in how a user likes or dislikes a particular movie.

### **QUESTION 10: k-NN collaborative lter**





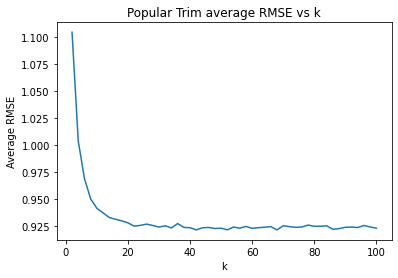
### **QUESTION 11:**

From the plot above, the average RMSE and MAE plateaued after k =20, so the **minimum k = 20.**

Best RMSE is 0.9723408317953796

Best MAE is 0.7475326132301008

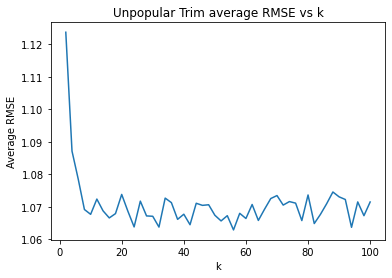
### **QUESTION 12:**



The minimum average RMSE is 0.9214390610885232

The popular testset seems to have better results than the whole testset, we think the reason could be since the set only contains movies with more than 2 ratings, these movies have more users’ ratings and therefore easier to be used to predict other users’ ratings.

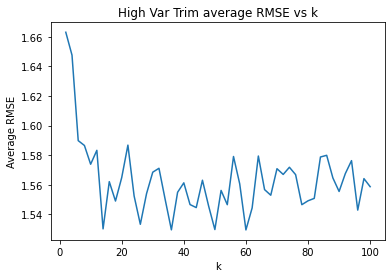
### **QUESTION 13:**



The minimum average RMSE is 1.0629163324177302

The unpopular testset seems to have worse results than the whole testset, we think the reason the set only contains movies with no greater than 2 ratings, and these movies do not have sufficient data or the rating data is too biased to predict other users’ ratings.

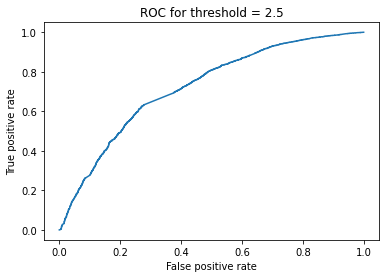
### **QUESTION 14:**



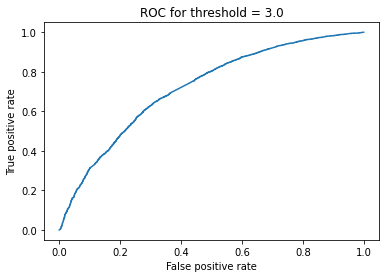
The minimum average RMSE is 1.5293858404053429

The high-variance testset has the worst results. The reason is this set of movies only contains “controversial” movies that users rate them quite differently, and when making predictions, the model will struggle to make correct predictions.

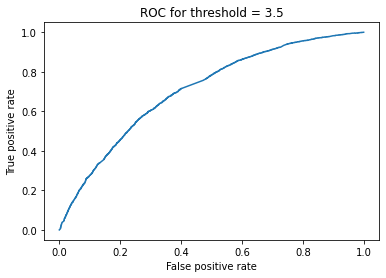
### **QUESTION 15:**



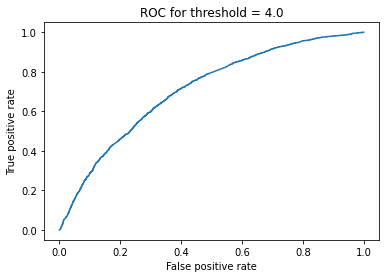
AUC is 0.7224195857284479



AUC is 0.7196189725215748



AUC is 0.7073335717600324

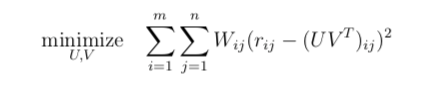


AUC is 0.7088318193766977

From the above ROC plots and its AUC, we can see that when the threshold is 2.5, AUC is maximum. From the data exploration, we know that most of the ratings received are 3 and above, and from the ROC results, we can induce that it is easier for the model to differentiate between rating 2.5 and others, than other thresholds. We can probably rationalize it as users rate movies 3 and above as they like it, but not a huge difference between 3 to 5; and when they rate 2.5 and below, they just dislike it, therefore there is a larger distinction between 2.5 and 3.

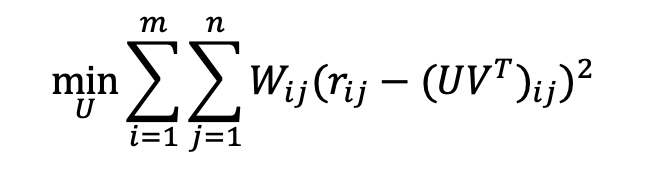
### **QUESTION 16:**

The optimization problem is given by equation 5:



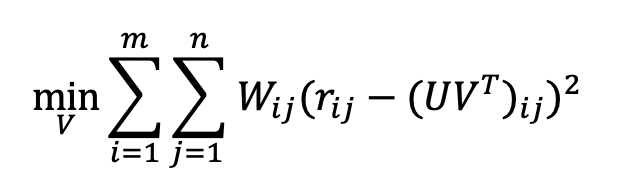
Is NOT convex, because both U & V are unknown variables, we can fix one of them in order to solve for the other one. When we fixed U or V, equation 5 becomes a least-squares problem and we can get the formulations:

When we fix V:



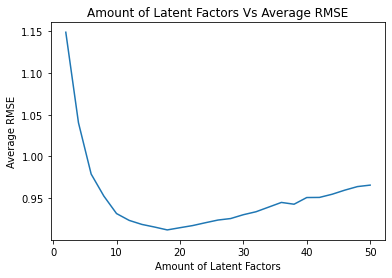
And in this example, when we fix V, solving for U, where V = V-hat, and it’s a fixed parameter, and U would be an independent variable.

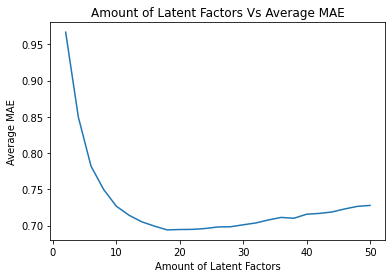
When we fixed U:



Vice versa, when we fix U, solving for V, where U = U-hat, and it’s a fixed parameter, and V would just be an independent variable.

### **QUESTION 17:**





In the plot, the prompt requested that we design an NNMF-based collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross-validation. Additionally, we were to 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. Lastly, as depicted above, the plot of the average RMSE (Y-axis) against k(X-axis) and the average MAE (Y-axis) against k (X-axis) was to be outputted. We imported from Surprise the matrix factorization packages, to prepare for the algorithm development.

And the result indicates that overall the MAE is lower than the average RMSE.

**QUESTION 18:**

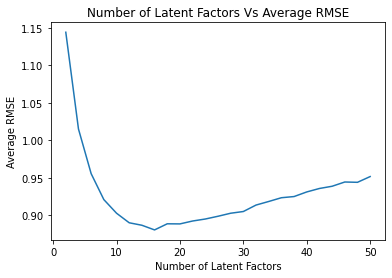
Minimum Average RMSE : 0.9121934685082319

Minimum Average MAE : 0.6940665001100047

**Best number of latent factors : 18**

In the above scenario, we were to use the plot from question 17 to find the optimal number of latent factors. Additionally, the optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. As illustrated above are the respective average RMSE and MAE. No. The optimal number of latent factors is greater than the number of movie genres.

### **QUESTION 19:**

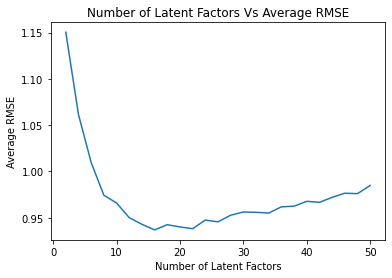


Minimum Average RMSE : 0.8805314259625033

Minimum Average MAE : 0.6716574915089666

In the above, we were tasked to design an NNMF collaborative filter to predict the rating of the movies in the popular movie trimmed test set and evaluate its performance using 10-fold cross-validation. Additionally, we had to 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. The above is the plot average RMSE (Y-axis) against (X-axis), as well as the minimum average RMSE.

### **QUESTION 20:**



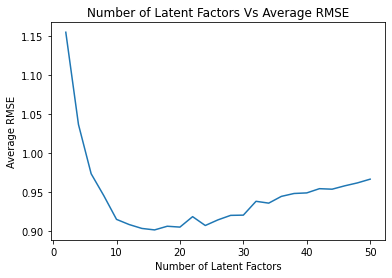
Minimum Average RMSE : 0.9368525311579339

Minimum Average MAE : 0.7128705107424866

In the above, we were tasked to 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. Additionally, we were to sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute average RMSE obtained by averaging the RMSE across all 10 folds. As depicted above, the plot average RMSE (Y-axis) against k (X-axis), as well the minimum average RMSE.

### 

### **QUESTION 21:**

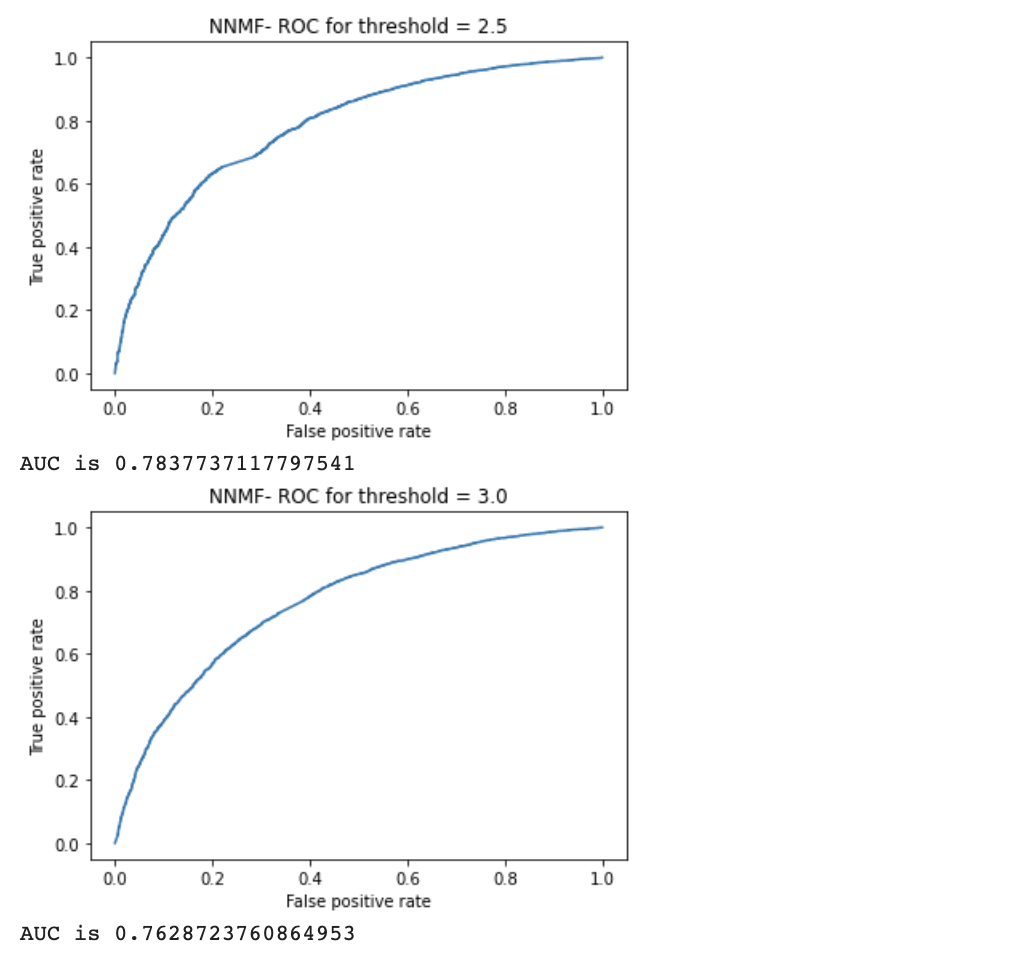


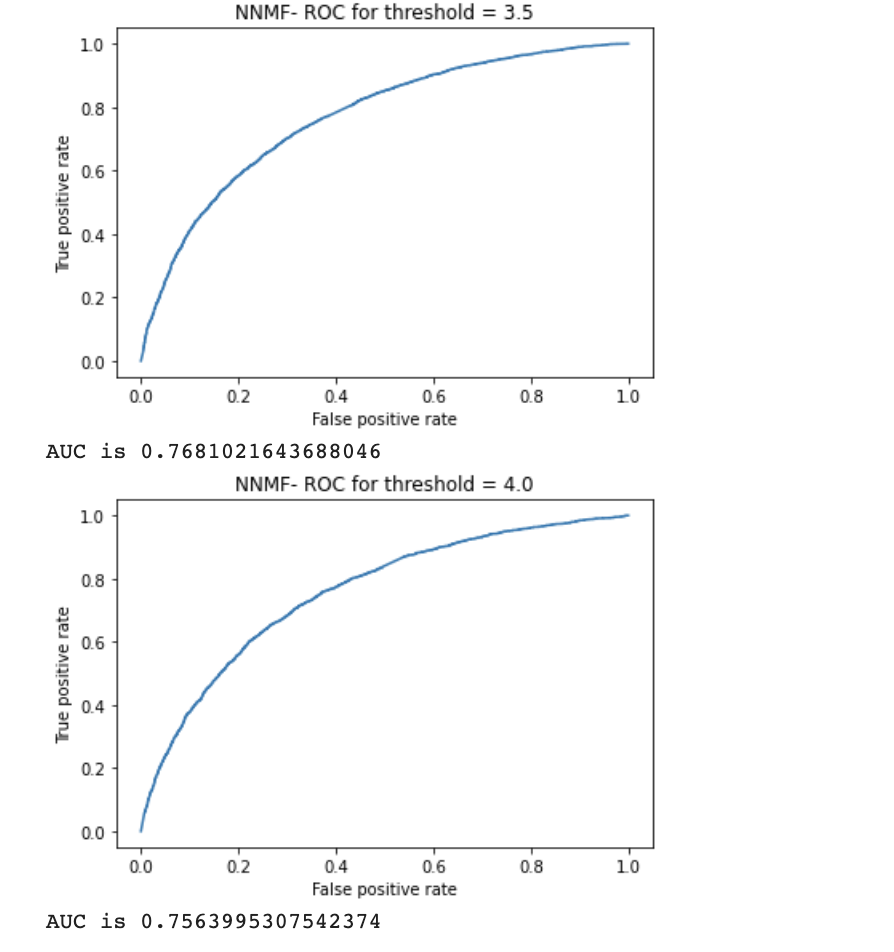
Minimum Average RMSE : 0.9019252097582285

Minimum Average MAE : 0.6810149668940758

In the above, we were tasked to 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 the 10-fold cross validation. Additionally we used sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k computed the average RMSE obtained by averaging the RMSE across all 10 folds. As depicted above, we plotted the average RMSE(Y-axis) against k(X-axis), as well the minimum average RMSE.

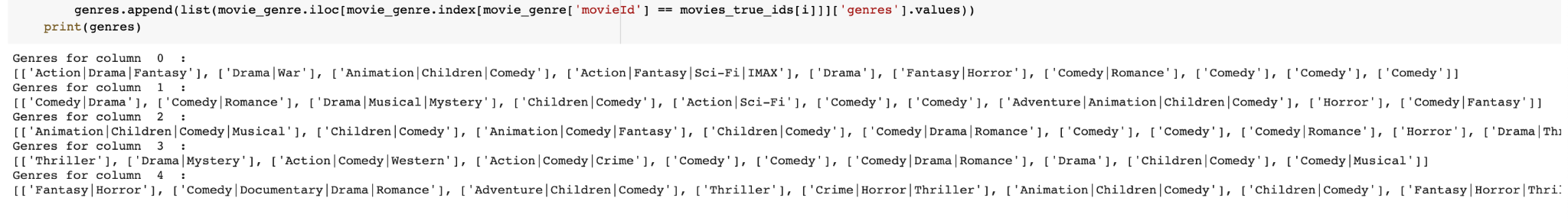
### **QUESTION 22:**

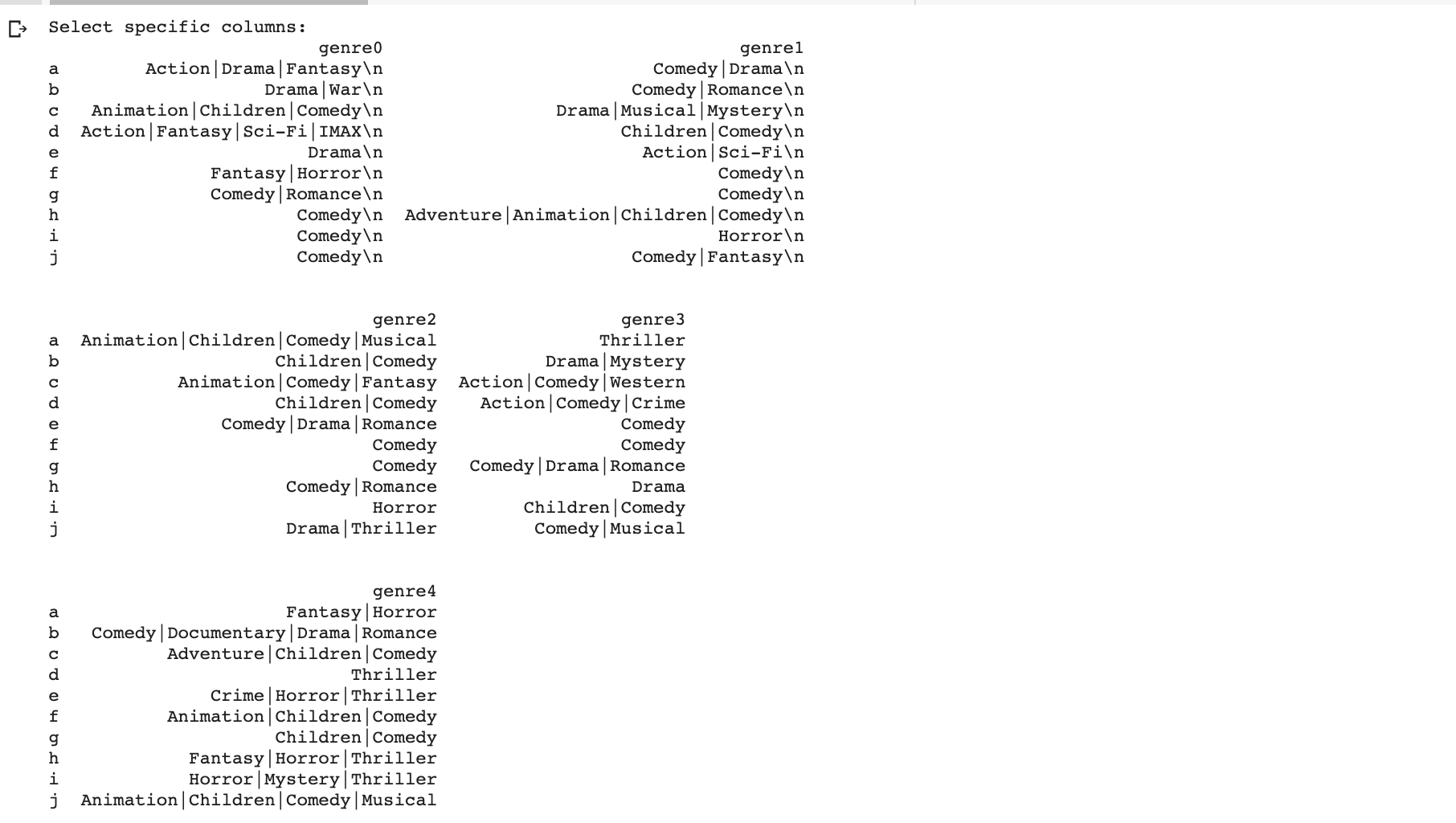




As depicted in the above, we were tasked to plot the ROC curves for the NNMF-based collaborative filter designed in question 17 for threshold values [2.5, 3, 3.5, 4]. Additionally, for the ROC plotting, we used the optimal number of latent factors found in question 18, as well the area under the curve (AUC) value(s).

### **QUESTION 23:**

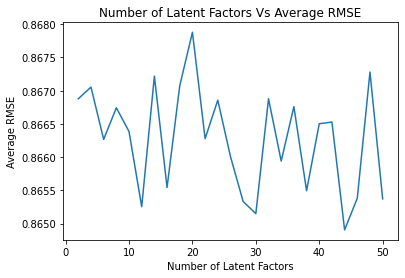


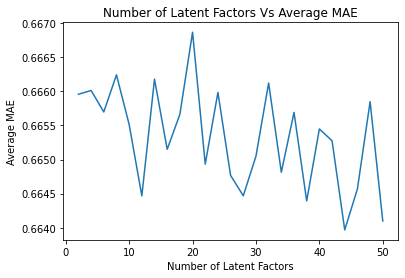


In the few examples of columns that were added here, each genre would have a more dominating or small collection of dominating genres. For example, in genre 1, you can see there’s a lot of Comedy and different subset of comedy; while in genre 2, it’s also comedy, but the collections of genres primarily focus on Children/Family related Comedy like children | comedy, or Fantasy comedy or Comedy Romance.

When we factorize a mxn matrix into two mxk and kxn matrices we are reducingyour "n"items to "k"factors. So maybe a factor means "Thriller ghosts", another factor might mean "Movies with a plot twist" etc. The key is that recommendation based on factors is more robust than just using movie similarities, maybe a user hasn’t seen "Matrix" but the user might have seen other movies that are related to Matrix via some latent factors and those can be used. In this specific case, there seems to have too much noise to directly point out the relationship between movie genres and latent factors; generally, the number of latent factors will influence how much nuance in the original utility matrix an algorithm can capture. So we should expect to see the movie genres cluster together more closely as the number of latent factors increases. But in every case, we are trying to find the optimal latent factor that will have optimal output while having relatively low computational cost. There is no one “best” latent factor.

### **QUESTION 24:**



****

For the above, we were tasked to 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. Additionally, we used sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and each k we computed the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. As depicted above, we outputted the plot of the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis).

### **QUESTION 25:**

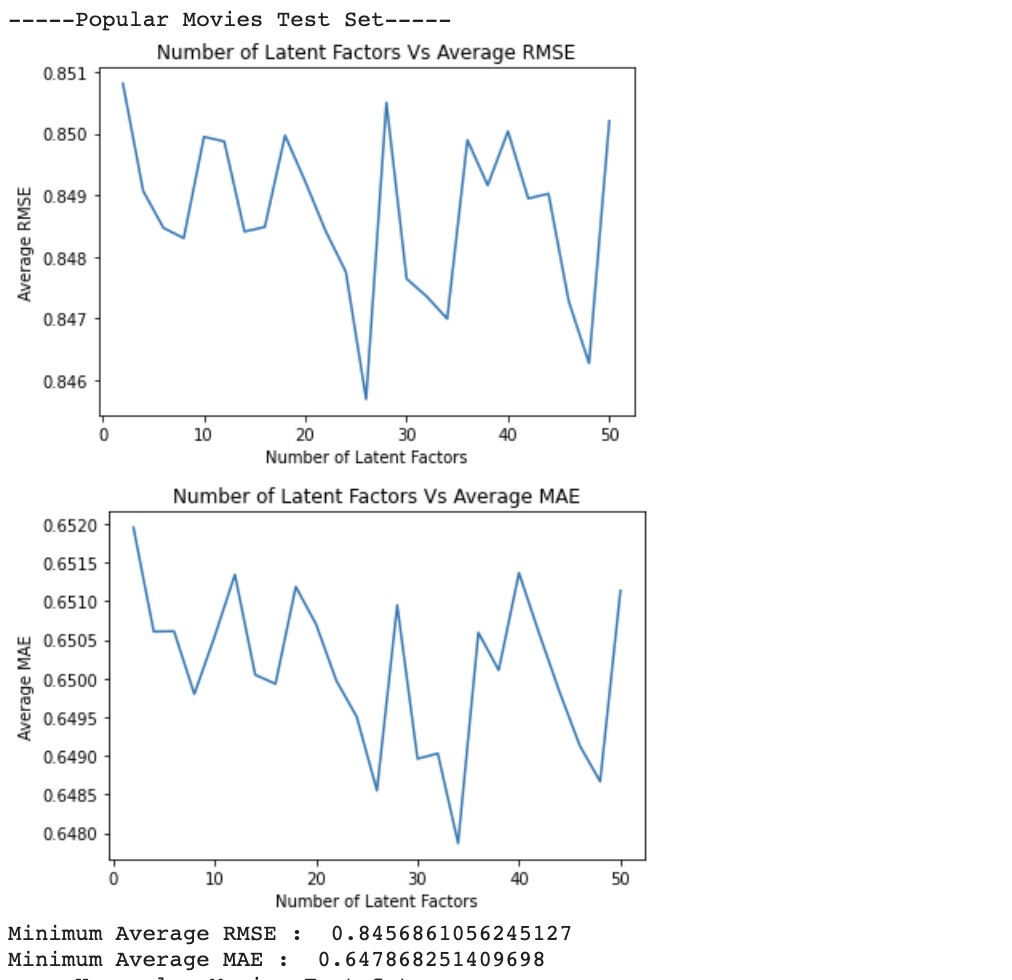
Minimum Average RMSE : 0.8649044938322493

Minimum Average MAE : 0.6639671858532243

**Best number of latent factors : 44**

In the above, we were tasked to use the plot from question 24, and to find the optimal number of latent factors. As informed by the prompt, the optimal number of latent factors is the value of k that gives the minimum average RMSE or the minimum average MAE. Also illustrated above is the minimum average RMSE and MAE.

### **QUESTION 26:**

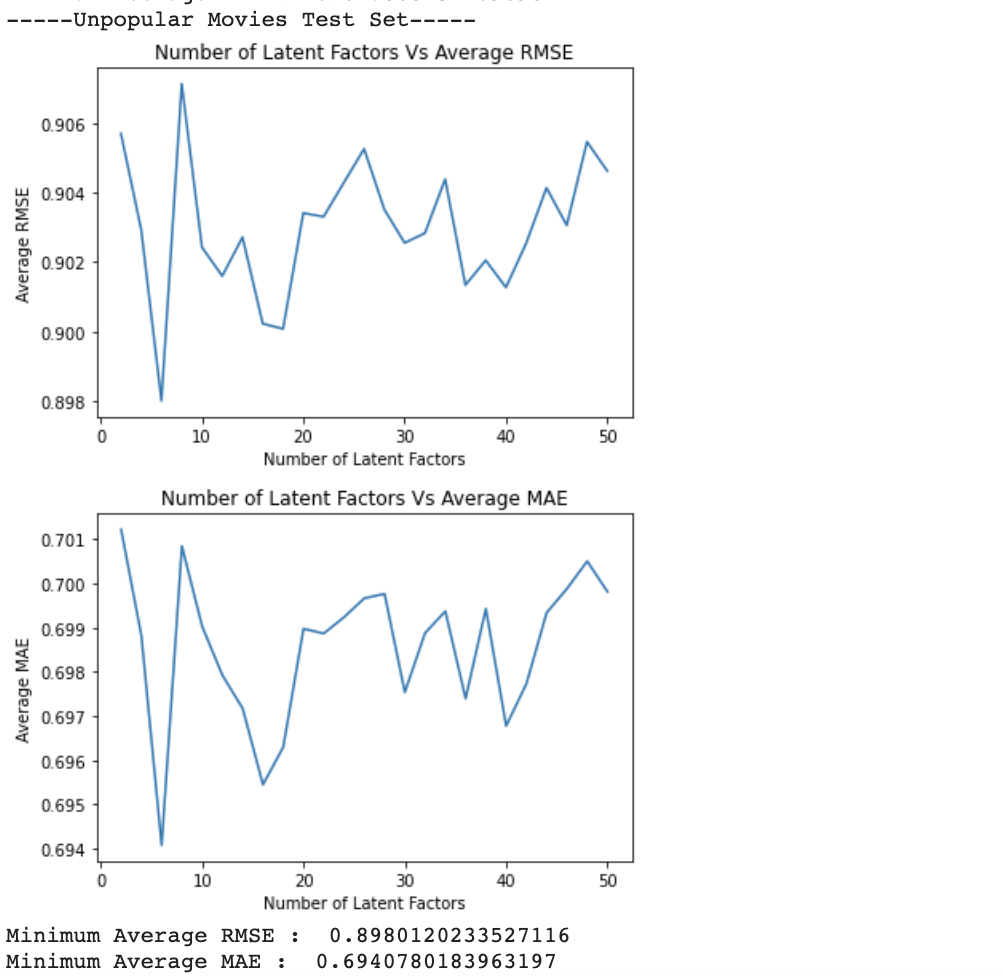


In the above, we were tasked to design a MF with bias collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it’s performance using 10-fold cross validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2 was used, and for each k we computed the average RMSE obtained by averaging the RMSE across all 10 folds. As illustrated above, we plotted the average RMSE (Y-axis) against k (X-axis). Additionally, the minimum average RMSE was reported as well.

### 

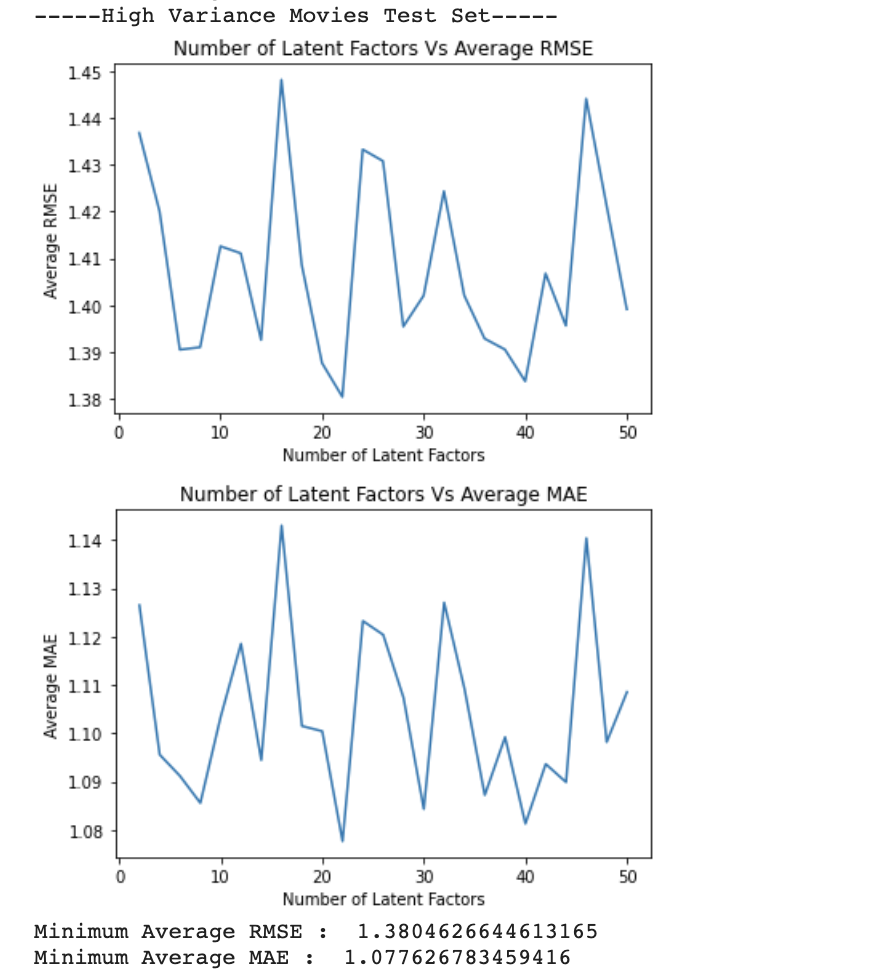
### 

### **QUESTION 27:**



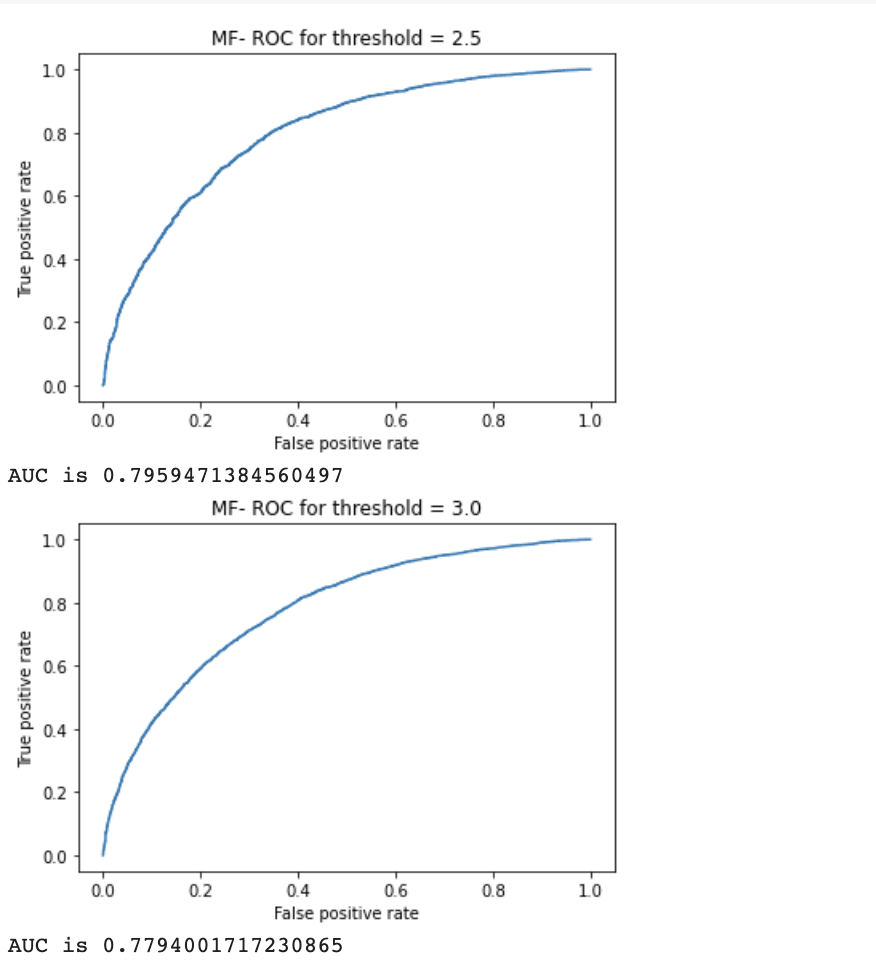
We were tasked to design a MF with bias collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate its performance using 10-fold cross validation. Sweep k (number of latent factors) was used from 2 to 50 in step sizes of 2 , and for each k we computed the average RMSE obtained by average the RMSE across all 10 folds. As illustrated above, we added the outputted plot average RMSE (Y-axis) against k (X-axis). Additionally, the minimum average RMSE was reported.

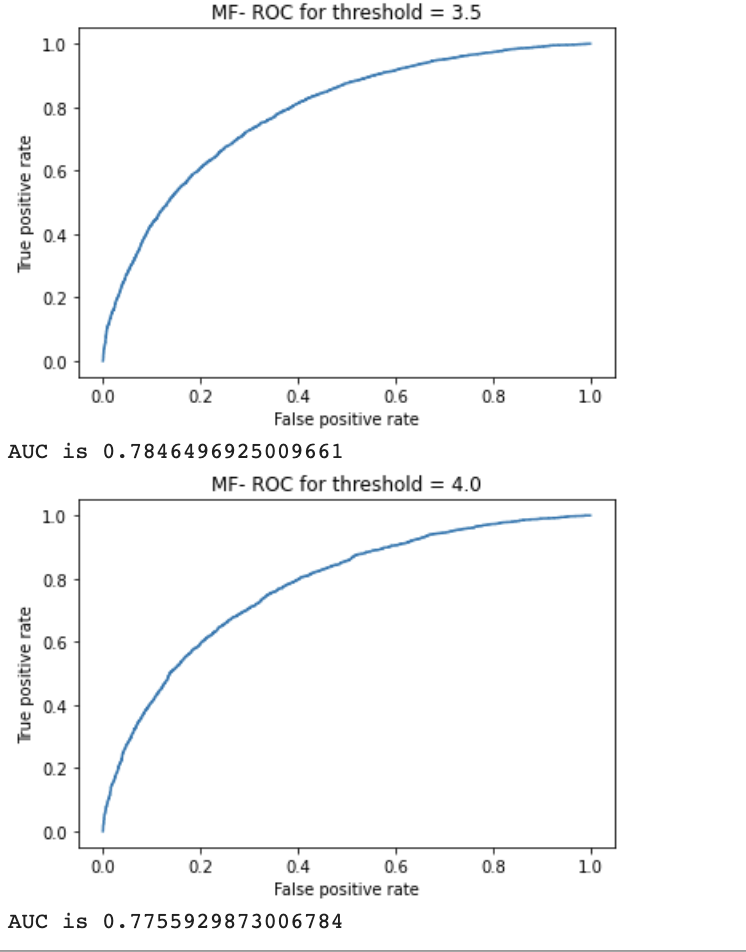
### **QUESTION 28:**



We were tasked to 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. We used sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k computed the average RMSE obtained by averaging the RMSE across all 10 folds. As depicted above, we plotted the average RMSE (Y-axis) against k (X-axis). Additionally, the minimum average RMSE was reported.

### **QUESTION 29:**





We were tasked to plot the ROC curves for the MF with bias collaborative filter designed in question 24 for threshold values [2.5,3,3.5,4]. Additionally, for the ROC plotting, we used the optimal number of latent factors found in question 25. Also reported in the area under the curve (AUC) value.

### **QUESTION 30:**

### The Average RMSE for this does not need to calculate miu-i for the training fold each time, so we can just use the single set of miu on the entire data set and validate, then divided it into the 10 validation folds. A naive collaborate filter here is used to predict rating in the dataset; the RMSE is calculated in each iteration and then the mean RMSE is calculated by dividing the # of validation folds.

Average RMSE on the whole data set: 0.9346861315371193

### **QUESTION 31:**

### Repeating the method on the popular-trimmed dataset and the final result is slightly lower than the mean RMSE in comparison to the whole dataset.

### Popular Movie trimAverage RMSE 0.9322983341685929

### **QUESTION 32:**

### Repeating the method on the unpopular-trimmed dataset and the final result is slightly higher than the mean RMSE in comparison to the whole dataset.

### Unpopular Movie trimAverage RMSE 0.9711056536885639

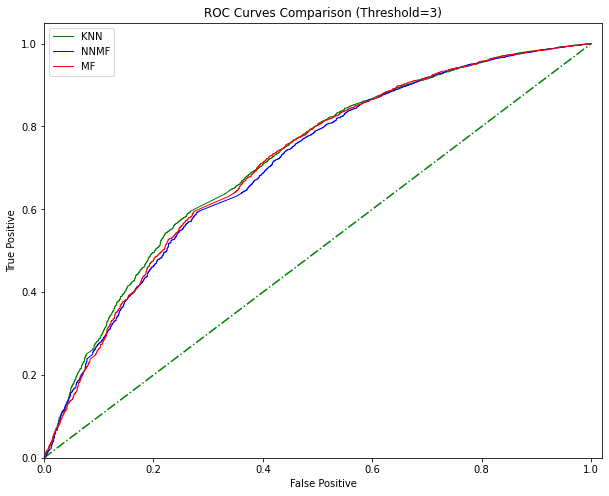
### **QUESTION 33:**

### Repeating the method on the high-variance-trimmed dataset and the final result is higher than the mean RMSE in comparison to the whole dataset and the other two trimming methods.

### High Variance Movie trimAverage RMSE 1.465782083175942

Seems like the Naive filtering works the best on the popular and unpopular trim data set, and not working as well on the high variance dataset.

### **QUESTION 34:**



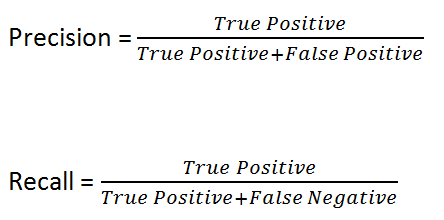
By seeing the performance comparison of the three methods, we can see that the results are very close, with barely significant differences between one another. MF & KNN seem to be slightly smoother than NNMF models, with the largest ROC with threshold = 3. With KNN being slightly smoother as well, this indicates that they are comparatively better models at predicting the ratings. Though, the differences are minimum based on the graph.

**QUESTION 35:**

Precision: The recommended items the user liked, divided by the items that got recommended to the user.

Recall: The recommended items the user liked, divided by all of the items the user liked (recommended or not)

Mathematically speaking:

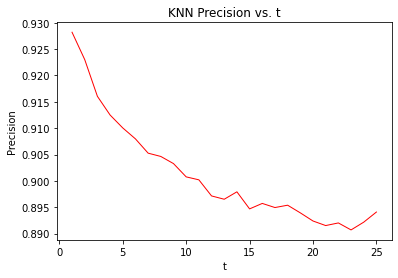


The precision is the correctness of the prediction, shows us how many positive predictions are actually positive, it measures the accuracy of positives.

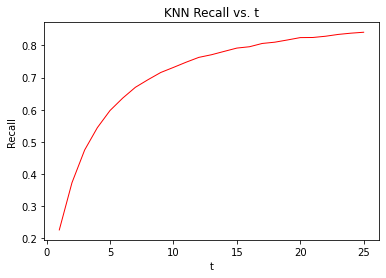
While recall tells us the completeness of the prediction, tells us the percentage of positive items were predicted correctly.

**QUESTION 36:**

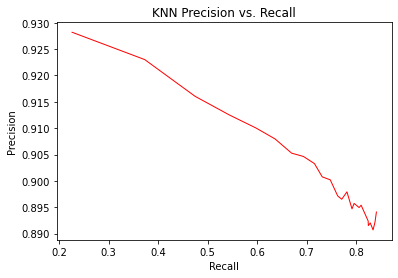
Best k = 20



As t increases, the Precision decreases, they have an inverse relationship. Since it would be simpler to predict the smaller amount of movies that the user would definitely like, as t increases, the precision would drop - as expected. Even then, the precision percentage stays above 0.89 in this graph.



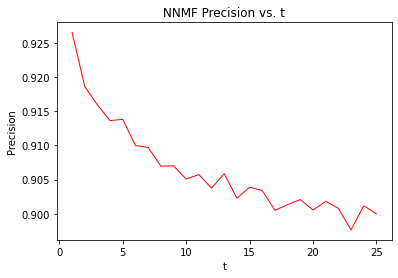
As t increases, the recall increase, they have a positive relationship. Seems like Recall is exponentially growing, as t increases.



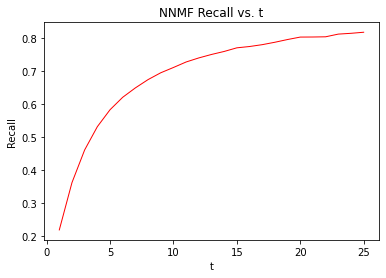
Precision and recall have an inverse relationship. As recall increases the precision value decrease. It indicates a trade-off between optimizing precision vs. recall.

### **QUESTION 37:**

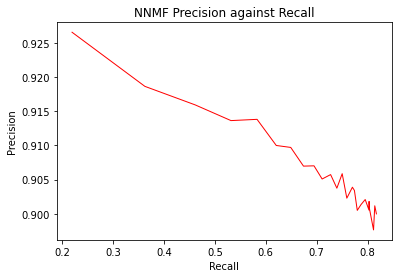
### Best n-factors = 18



As t increases, the Precision decreases, they have an inverse relationship. Since it would be simpler to predict the smaller amount of movies that the user would definitely like, as t increases, the precision would drop - as expected. Even then, the precision percentage stays above 0.89 in this graph.



As t increases, the recall increase, they have a positive relationship. Seems like Recall is exponentially growing, as t increases.

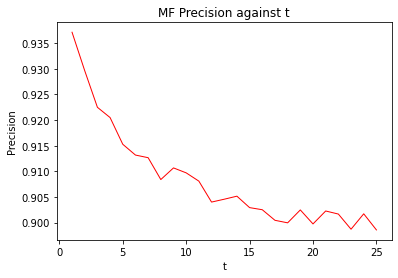


Precision and recall have an inverse relationship. As recall increases the precision value decrease. It indicates a trade-off between optimizing precision vs. recall.

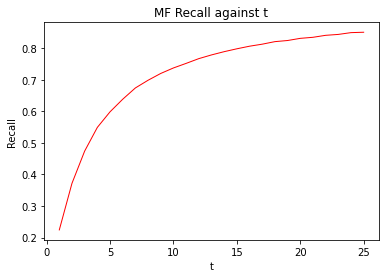
### 

### **QUESTION 38:**

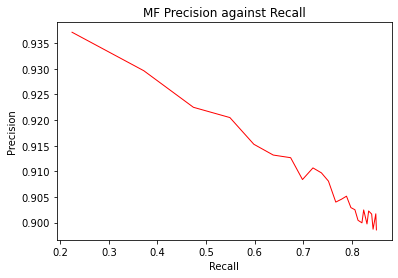
best n\_factor = 44



As t increases, the Precision decreases, they have an inverse relationship. Since it would be simpler to predict the smaller amount of movies that the user would definitely like, as t increases, the precision would drop - as expected. Even then, the precision percentage stays above 0.90 in this graph.



As t increases, the recall increase, they have a positive relationship. Seems like Recall is exponentially growing, as t increases.



Precision and recall have an inverse relationship. As recall increases the precision value decrease. It indicates a trade-off between optimizing precision vs. recall.

### 

### **QUESTION 39:**

This graph shows the precision-recall curve obtained in questions 36-38, with color-coded lines. From the above plot, clearly indicates that for average recall MF is a better recommendation algorithm for the MOVIELENS dataset than the other two. The three algorithms all show the inverse relationship between precision and recall.

[1] <http://files.grouplens.org/datasets/movielens/ml-latest-small.zip>