Project_3

February 14, 2022

1 Project 3 - Recommender Systems

Gorkem Camli (105709280)

Library imports

1.1 Dataset

Data Exploration

```
Ratings Raws length:  100836
Data preview:

   userId  movieId  rating   timestamp
0     496   112852     3.0  1415520462
1     391     1947     4.0  1030945141
2     387     1562     1.5  1095041022
3     474     2716     4.5  1053020930
4     483    88125     4.5  1311337237

Number of users: 610
Number of movies: 9724
```

1.1.1 Question 1

Question 1.A Compute the sparsity of the movie rating dataset:

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

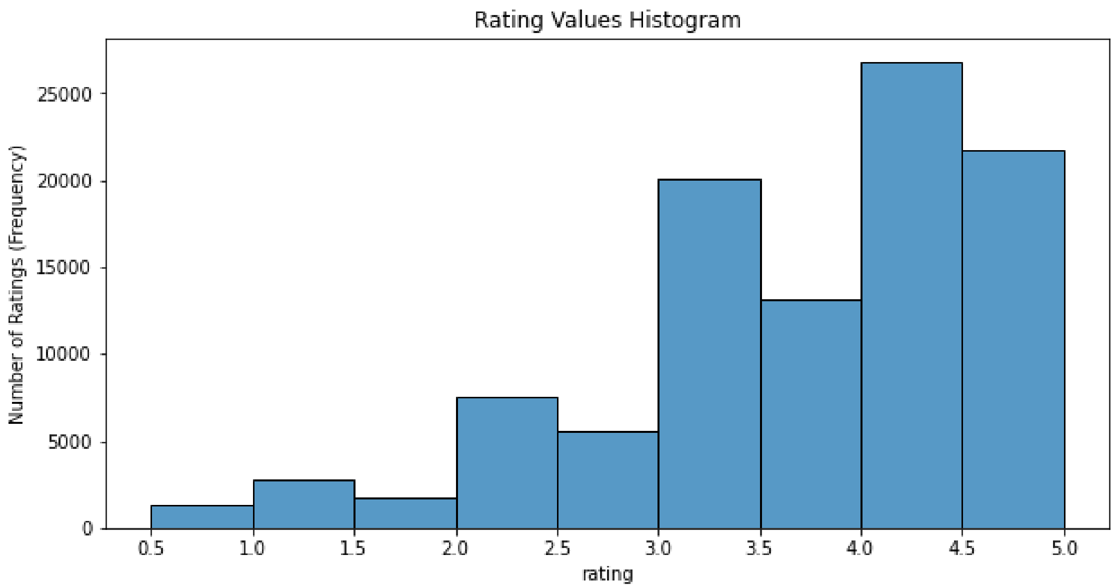
```
Total number of users: 610
Total number of movies: 9724
Total number of available ratings: 100836
Total number of possible ratings = number of movies x number of users = 5931640


The sparsity of the movie rating dataset is: 0.016999683055613623
```

Question 1.B Plot a histogram showing the frequency of the rating values: Bin the raw 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 that fall within each bin and use this count as the height of the vertical axis for that particular bin. Comment on the shape of the histogram.

```
Ratings Scale:
[0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
```

```
 Text(0, 0.5, 'Number of Ratings (Frequency)')
```



The ratings are within the 0.5 to 5.0 range. Users who gave ratings tends to give higher rating scores since the trend of the histogram is towards the higher ratings. Around 81% of the ratings are 3 and above and 21.5% is 4.5 and above. One possible interpretation is that users may tend to give ratings to the movies they like or they liked the movies they watched.

```
Ratings Count:
 4.0    26816
3.0    20046
5.0    13211
3.5    13136
4.5     8553
2.0     7551
2.5     5551
1.0     2811
1.5     1791
0.5     1370
Name: rating, dtype: int64
Ratings Count normalized:
 4.0    0.265937
```

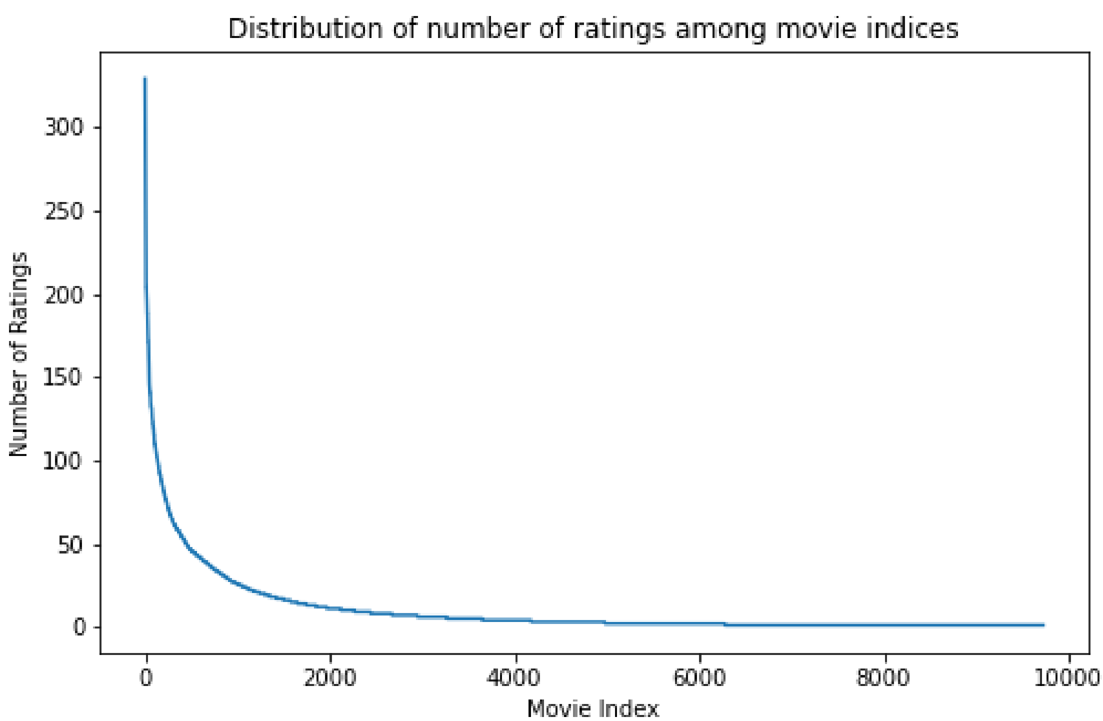
```
3.0    0.198798
5.0    0.131015
3.5    0.130271
4.5    0.084821
2.0    0.074884
2.5    0.055050
1.0    0.027877
1.5    0.017762
0.5    0.013586
Name: rating, dtype: float64
```

Question 1.C Plot the distribution of the number of ratings received among movies: 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; ties can broken in any way. A monotonically decreasing trend is expected.

```
Top 10 movies with most ratings

                 0    1    2    3     4    5    6    7    8    9
movieId        356  318  296  593  2571  260  480  110  589  527
rating_count   329  317  307  279   278  251  238  237  224  220
```

```
 Text(0, 0.5, 'Number of Ratings')
```



```
              count          mean           std  min     25%     50%  \
movieId      9724.0  42245.024373  52191.137320  1.0  3245.5  7300.0
rating_count 9724.0     10.369807     22.401005  1.0     1.0     3.0

                  75%       max
movieId      76739.25  193609.0
rating_count     9.00     329.0
```

```
Rating Count Normalized Value Counts:
```

```
1      0.354381
2      0.133484
3      0.082271
4      0.054504
5      0.039284
         …
203    0.000103
211    0.000103
251    0.000103
215    0.000103
307    0.000103
Name: rating_count, Length: 177, dtype: float64
```

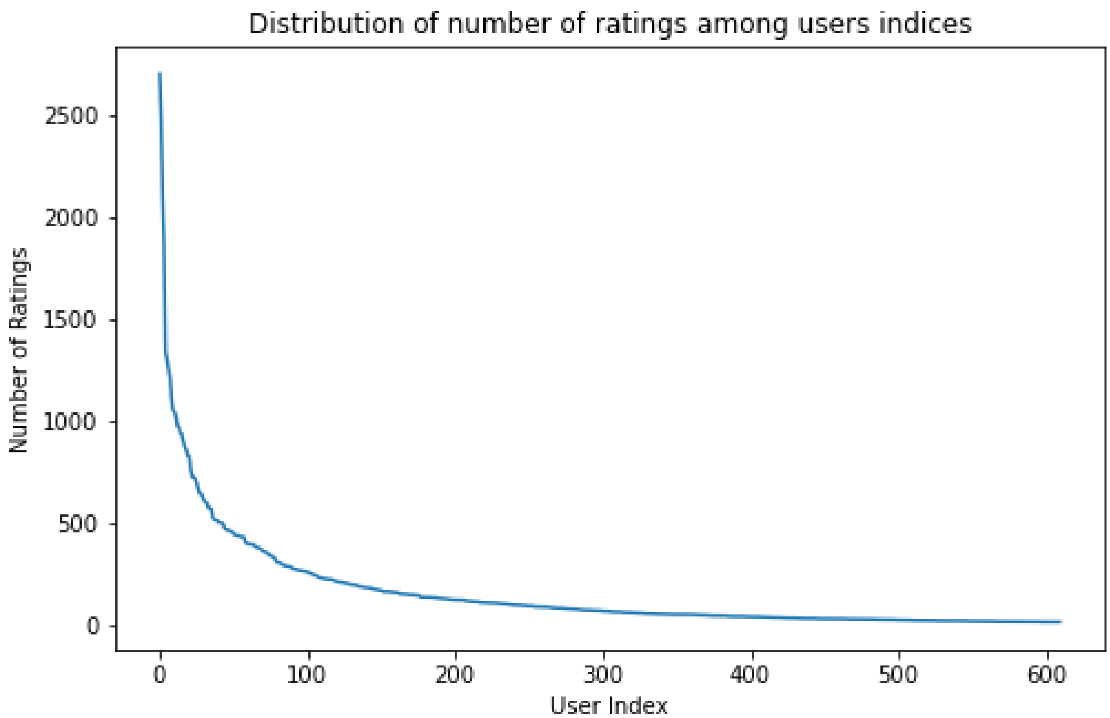
As expected we observe a monotonically decreasing trend. Majority of the movies rated very few or 1 times only. This shows that there a few popular movies that are rated among many users and majority of the movies rated by few users. This points out that there is sparsity in the data.

Question 1.D Plot the distribution of ratings among users: The X-axis should be the user index ordered by decreasing frequency and the Y -axis should be the number of movies the user has rated. The requirement of the plot is similar to that in Question C.

```
Top 10 users with most ratings
```

	0	1	2	3	4	5	6	7	8	9
userId	414	599	474	448	274	610	68	380	606	288
rating_count	2698	2478	2108	1864	1346	1302	1260	1218	1115	1055

```
Text(0, 0.5, 'Number of Ratings')
```




```
               count        mean         std   min     25%    50%     75%  \
userId         610.0  305.500000  176.236111   1.0  153.25  305.5  457.75
rating_count   610.0  165.304918  269.480584  20.0   35.00   70.5  168.00

                  max
userId          610.0
rating_count   2698.0
```

```
Rating Count Normalized Value Counts:

20      0.022951
21      0.024590
22      0.022951
23      0.021311
24      0.011475
          …
1346    0.001639
1864    0.001639
2108    0.001639
2478    0.001639
2698    0.001639
Name: rating_count, Length: 261, dtype: float64

Rating Count Normalized Value Counts after cumsum from lines 45-51:
```

```
66    0.483607
67    0.486885
68    0.488525
69    0.496721
70    0.500000
71    0.501639
Name: rating_count, dtype: float64
```

Again we see a decreasing trend with a tailed curve. There are some users who did a lot of ratings but majority seems to rate low number of movies. The minimum number of ratings among the users is 20. Half of the users rated 70 or less movies out of 9724 movies, which again shows that there is an uneven distribution and sparsity in the ratings given by users.

Question 1.E Discuss the salient features of the distributions from Questions C,D and their implications for the recommendation process.

In 1.C, the number of ratings given to movies has a monotonically decreasing trend with a long tail on the right. This means majority of the movies get very low number of ratings where few of the movies get majority of the ratings. Around 35.4% movies get only 1 rating. Similarly, in 1.D, the number of ratings given by users has a monotonically decreasing trend with a long tail as well. This means majority of the users gave few ratings, and there are few users with very high number of ratings. Half of the users gave at least 20 and at most 70 movie ratings. Given that we have 610 users and 9724 movies, the ratings matrix R is a sparse matrix. The sparsity we calculate in Q1.A with 0.0169 confirms this. Having a sparse matrix makes recommendation systems job harder, as there are few links between users and movies. This means if there is a low number rating for a specific movie, it will be harder to rate that movie based on similar users, or if the user has a low number of rating it will be hard to learn about what users might like or not. Given the sparsity of matrix finding the similarities between users or movies will be challenging because we will have to work on small number of ratings, given that majority of the values are 0. This might cause problems such as overfitting or false predictions on the ratings. As described in the page 6 of the specification part, we might need to regularize our models to avoid overfitting.

Question 1.F Compute the variance of the rating values received by each movie: 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 resulting histogram.

```
Movie based stats table preview:

                        0           1          2         3          4  \
movieId           1.00000    2.000000   3.000000  4.000000   5.000000
max_rating        5.00000    5.000000   5.000000  3.000000   5.000000
min_rating        0.50000    0.500000   0.500000  1.000000   0.500000
rating_variance   0.69699    0.777419   1.112651  0.726190   0.822917
rating_count    215.00000  110.000000  52.000000  7.000000  49.000000
rating_mean       3.92093    3.431818   3.259615  2.357143   3.071429

                         5          6        7       8           9
```

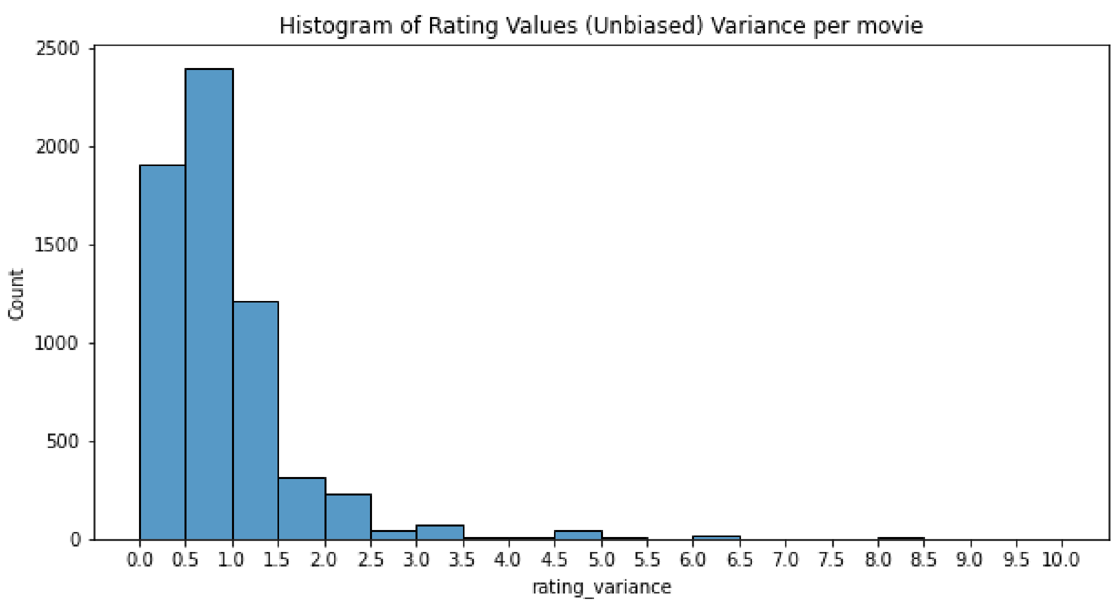
movieId	6.000000	7.000000	8.000000	9.000	10.000000
max_rating	5.000000	5.000000	5.000000	5.000	5.000000
min_rating	1.000000	1.000000	1.000000	1.500	0.500000
rating_variance	0.670841	0.955625	1.267857	0.950	0.738535
rating_count	102.000000	54.000000	8.000000	16.000	132.000000
rating_mean	3.950980	3.185185	2.875000	3.125	3.496212

Description of Movie based stats table:

	movieId	max_rating	min_rating	rating_variance	rating_count \
count	9724.000000	9724.000000	9724.000000	6278.000000	9724.000000
mean	42245.024373	3.912999	2.416495	0.857169	10.369807
std	52191.137320	1.056532	1.241600	0.795839	22.401005
min	1.000000	0.500000	0.500000	0.000000	1.000000
25%	3245.500000	3.500000	1.500000	0.395833	1.000000
50%	7300.000000	4.000000	2.500000	0.702111	3.000000
75%	76739.250000	5.000000	3.500000	1.105310	9.000000
max	193609.000000	5.000000	5.000000	10.125000	329.000000

	rating_mean
count	9724.000000
mean	3.262448
std	0.869874
min	0.500000
25%	2.800000
50%	3.416667
75%	3.911765
max	5.000000

Text(0.5, 1.0, 'Histogram of Rating Values (Unbiased) Variance per movie')



```
Movies percentage that has variance less than or equal 1: 44.94%
Movies percentage that has variance less than or equal 1.5: 56.78%
```

I used unbiased variance to calculate variance values. From the histogram, we can see that majority of the movies variance are in the range of 0 to 1.5. almost 44.94% of the movies has a variance <= 1 and 56.78% movies has a variance of <=1.5. This means that most of the movies min and max rating range is close to each other, and the ratings given by users are similar and could be relied for the recommendations. There are few movies where the variance range is very high and even extreme cases where variance is more than 5. From our observation those movies are the ones that received few ratings and the ratings are very different. For example for the movieId 2068, there are 2 ratings: [5.0, 0.5] - variance is 10.125. For these few extreme cases and movies with high variances, we can commment that the ratings given by users are inconsistent and not very reliable.

1.2 Neighborhood-based collaborative filtering

1.2.1 Question 2

Question 2.A Write down the formula for μ_u in terms of I_u and r_{uk}

$$\mu_u = \frac{\text{Sum of ratings given by user u}}{\text{Number of ratings given by user u}}$$

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

μ_u represents the mean of the ratings given by user u.

Question 2.B In plain words, explain the meaning of

$$I_u \cap I_v \tag{1}$$

. Can

$$I_u \cap I_v = \emptyset \tag{2}$$

? (Hint: Rating matrix R is sparse)

$$I_u \cap I_v \tag{3}$$

are the set of movies where both user u and user v rated. Because R matrix is sparse, it is very likely that for some user u and user v, they don't have any common movies they rated, hence the intersection of them is empty:

$$I_u \cap I_v = \emptyset \tag{4}$$

.

1.2.2 Question 3

Understanding the Prediction function: 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.)

Ratings can be subjective for every person, some people might be more prone to give higher scores and some prone to give lower scores. Therefore, score distribution of users also could be very different. We are not interested in the absolute rating for a movie from a user, because let's say an average rating for user A is 4 and he gave a movie rating 2.5 to movie X, thinks movie is bad, and user B has an average rating of 2. Then when we try to predict User B's rating, if we use absolute score, user B will think like user A liked the movie (user B's avg was 2) and become prone to like movie X. If both users have similar tastes, B should have dislike the movie as well. Similarly these things can happen when there are users who rate high or low all the time. This cause biases during the prediction of ratings.

To reduce subjectivity and bias, we should interpret the score given by users relatively rather than absolute rating, and standardize them. Otherwise we might end up biasing our predicted rating and misinterpreting the score given by other users. To standardize user ratings we do mean centering around each user. In this way, mean centering can reduce the user bias, and help us predict more accurate and unbiased rating scores.

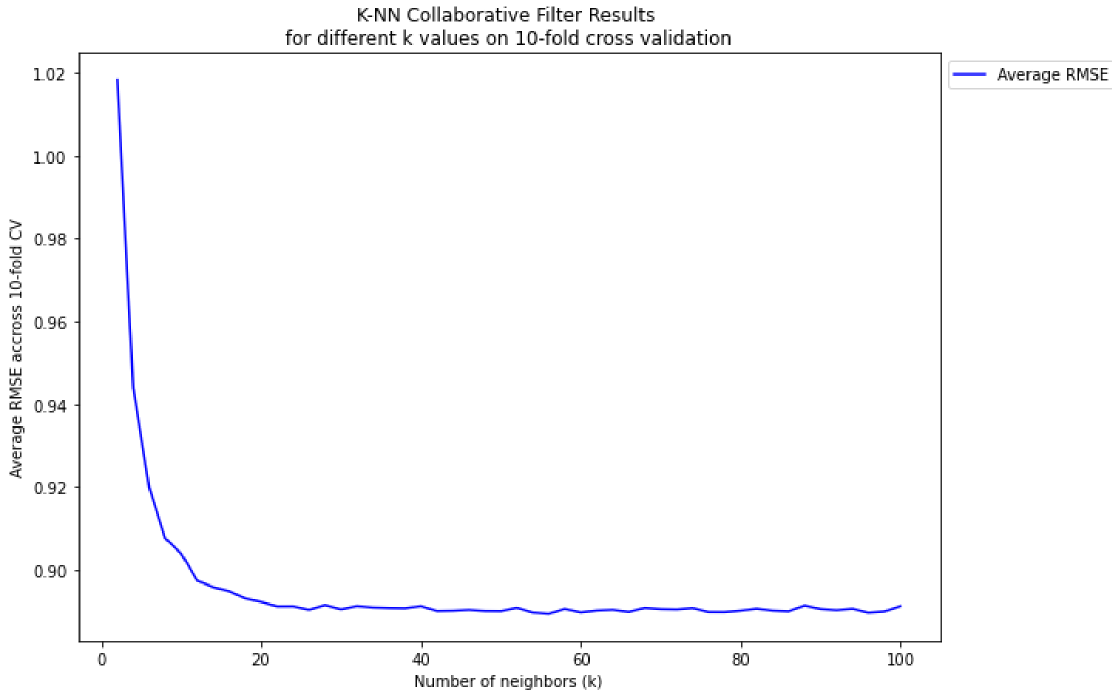
1.2.3 Question 4

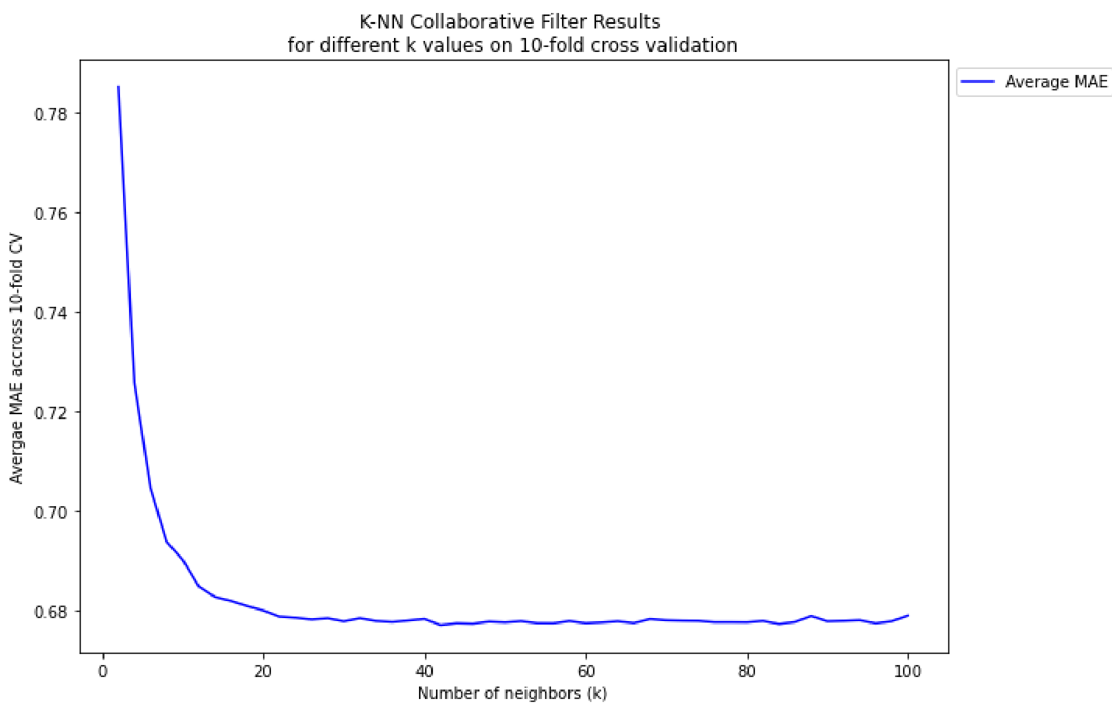
Design a k-NN collaborative filter to predict the ratings of the movies in the original dataset and evaluate its performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the 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).

I used KNNwithMeans model from surprise library to for the KNN collaborative filter because the prediction calculation made by KNNwithMeans was the one corresponding to Equation 3 in the project description pdf. The plots below shows that Average RMSE and MAE results accross 10-fold for each k from 2 to 100, with step sizes 2.

RMSE: Root Mean Square Error, takes the square root of average of the squared differences betweeen true ratings and predicted ratings. RMSE penalize more when the difference between true and predicted rating is higher (takes square of the error). Therefore, it is affected more by bad predictions and outliers.

MAE: Mean Average Error, is the average of absolute difference between true rating and predicted rating. Therefore, it is not biased towards big errors (bad predictions) as RMSE and it penalizes all predictions equally.


K-NN Collaborative Filter Results
for different k values on 10-fold cross validation
Average RMSE
Average RMSE accross 10-fold CV
Number of neighbors (k)
1.02
1.00
0.98
0.96
0.94
0.92
0.90
0
20
40
60
80
100


K-NN Collaborative Filter Results
for different k values on 10-fold cross validation
Average MAE
Avergae MAE accross 10-fold CV
Number of neighbors (k)
0.78
0.76
0.74
0.72
0.70
0.68
0
20
40
60
80
100

```
                                                                0   \
k                                                               2
avg_rmse                                                   1.0184
avg_mae                                                  0.785199
cv_results  {'test_rmse': [1.0064455149967226, 1.031740256…

                                                                1   \
k                                                               4
avg_rmse                                                 0.943863
avg_mae                                                  0.725791
cv_results  {'test_rmse': [0.9361455127443382, 0.936461550…

                                                                2   \
k                                                               6
avg_rmse                                                 0.919774
avg_mae                                                  0.704725
cv_results  {'test_rmse': [0.9167145310841434, 0.912939917…

                                                                3   \
k                                                               8
avg_rmse                                                 0.907544
avg_mae                                                  0.693875
cv_results  {'test_rmse': [0.8955187655975912, 0.896143192…

                                                                4   \
k                                                              10
avg_rmse                                                 0.903836
avg_mae                                                  0.690233
cv_results  {'test_rmse': [0.9004002958024331, 0.906640460…

                                                                5   \
k                                                              12
avg_rmse                                                 0.897336
avg_mae                                                  0.684864
cv_results  {'test_rmse': [0.8920181051105353, 0.904678838…

                                                                6   \
k                                                              14
avg_rmse                                                 0.895703
avg_mae                                                  0.682759
cv_results  {'test_rmse': [0.8984940103116476, 0.884587799…

                                                                7   \
k                                                              16
avg_rmse                                                 0.894728
avg_mae                                                   0.68196
cv_results  {'test_rmse': [0.8944573838012619, 0.894354533…
```

```
                                                           8  \
k                                                         18
avg_rmse                                            0.892995
avg_mae                                             0.680934
cv_results  {'test_rmse': [0.8964935529488661, 0.891298673…

                                                           9  \
k                                                         20
avg_rmse                                            0.892148
avg_mae                                             0.680044
cv_results  {'test_rmse': [0.8999086243769336, 0.887705797…

                                                          10  \
k                                                         22
avg_rmse                                            0.891058
avg_mae                                             0.678801
cv_results  {'test_rmse': [0.8803154535306453, 0.903646226…

                                                          11  \
k                                                         24
avg_rmse                                            0.891069
avg_mae                                             0.678574
cv_results  {'test_rmse': [0.8995998966443965, 0.891611505…

                                                          12  \
k                                                         26
avg_rmse                                            0.890236
avg_mae                                             0.678228
cv_results  {'test_rmse': [0.8849716486009225, 0.897649236…

                                                          13  \
k                                                         28
avg_rmse                                            0.891363
avg_mae                                             0.678474
cv_results  {'test_rmse': [0.8772238268582336, 0.891535332…

                                                          14
k                                                         30
avg_rmse                                            0.890353
avg_mae                                             0.677846
cv_results  {'test_rmse': [0.8844422049951735, 0.891825697…
```

From the above plot, we can observe that both RMSE and MAE curves start with high error values, follow a decreasing trend and plateus after certain k. As expected, RMSE Errors is higher than MAE. At first by increasing k values, we decrease the error a lot, probably because the neighbors we look at are not enough to do correct predictions, and as we increase k values, the model starts to perform better as it learns from more neighbors. However, after a moment increasing k doesn't change our results too much and the errors stabilizes. The reeason for that might be the new added neighbors, become less similar to the user, which causes the Pearson Correlation to be small and the new neighbors doesn't effect too much the rating prediction (Equation 3).

1.2.4 Question 5

Use the plot from question 4, 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.

	k	avg_rmse	avg_mae
0	2	1.018401	0.785199
1	4	0.943863	0.725791
2	6	0.919774	0.704725
3	8	0.907544	0.693875
4	10	0.903836	0.690233
5	12	0.897336	0.684864
6	14	0.895703	0.682759
7	16	0.894728	0.681960
8	18	0.892995	0.680934
9	20	0.892148	0.680044
10	22	0.891058	0.678801
11	24	0.891069	0.678574
12	26	0.890236	0.678228
13	28	0.891363	0.678474
14	30	0.890353	0.677846
15	32	0.891130	0.678485
16	34	0.890795	0.677920
17	36	0.890675	0.677728
18	38	0.890613	0.678016
19	40	0.891130	0.678335
20	42	0.889944	0.677095
21	44	0.890013	0.677464
22	46	0.890212	0.677373
23	48	0.889950	0.677807
24	50	0.889922	0.677653
25	52	0.890722	0.677890
26	54	0.889569	0.677479
27	56	0.889263	0.677460
28	58	0.890463	0.677916
29	60	0.889603	0.677471
30	62	0.890063	0.677636
31	64	0.890237	0.677866
32	66	0.889732	0.677505
33	68	0.890697	0.678320
34	70	0.890405	0.678065

35	72	0.890330	0.677988
36	74	0.890654	0.677950
37	76	0.889695	0.677687
38	78	0.889682	0.677690
39	80	0.890029	0.677669
40	82	0.890504	0.677934
41	84	0.890039	0.677317
42	86	0.889870	0.677732
43	88	0.891276	0.678926
44	90	0.890423	0.677864
45	92	0.890149	0.677949
46	94	0.890485	0.678085
47	96	0.889529	0.677449
48	98	0.889831	0.677874
49	100	0.891120	0.679005

After checking the plot in Question 4, both RMSE and MAE curves starts to plateau somewhere close to k=20. When we check above table to further investigate, we can see 2 points where the errors starts to stabilizes after reaching k=12, both RMSE and MAE results loses its momentum to decrease, and changes starts to happen in 3rd decimal points. However, if we look further another decrease happens in k=22, then the error starts to change mostly around 4th decimal point, and stays in a steady state until the end of the experiment.

Errors when k=22 - RMSE = 0.891058 - MAE = 0.678801

I chose minimum k as k=22 for KNN.

1.2.5 Question 6

For EACH of the 3 subsets in the test set, design:

A k-NN collaborative filter to predict the ratings of the movies in the test subset (i.e Popular, Unpopular or High-Variance) and evaluate each of the three models' 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. - Plot the ROC curves for the k-NN collaborative filters for threshold values [2.5, 3, 3.5, 4]. For each of the plots, also report the area under the curve (AUC) value.

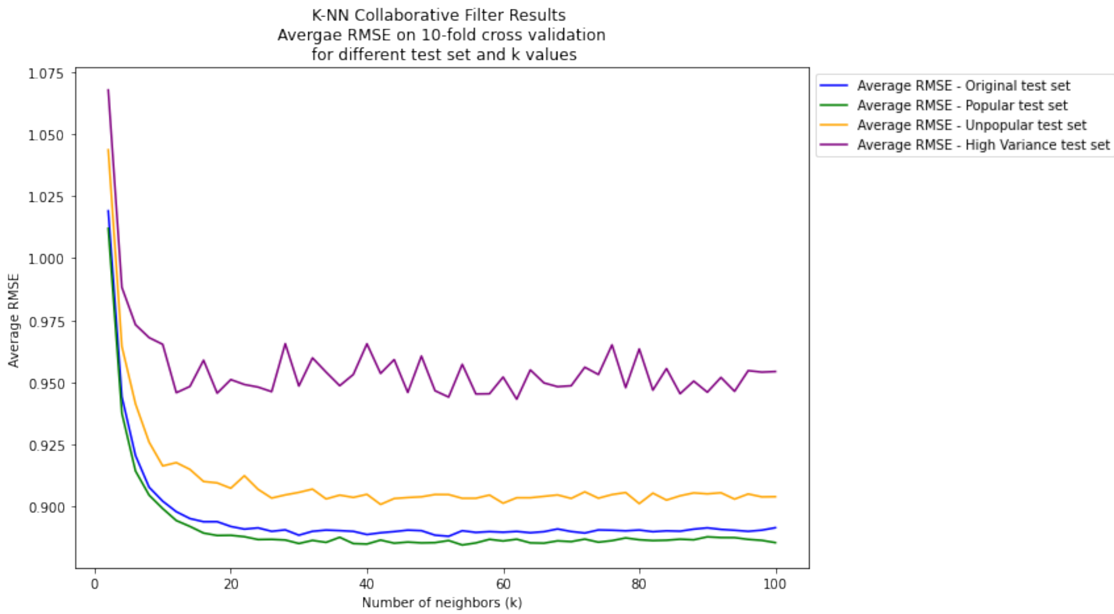
I first created 3 movie id lists to be used in the questions where we create custom test set: popular movies, unpopular movies and high variance movies following the rules given for each trimming type.

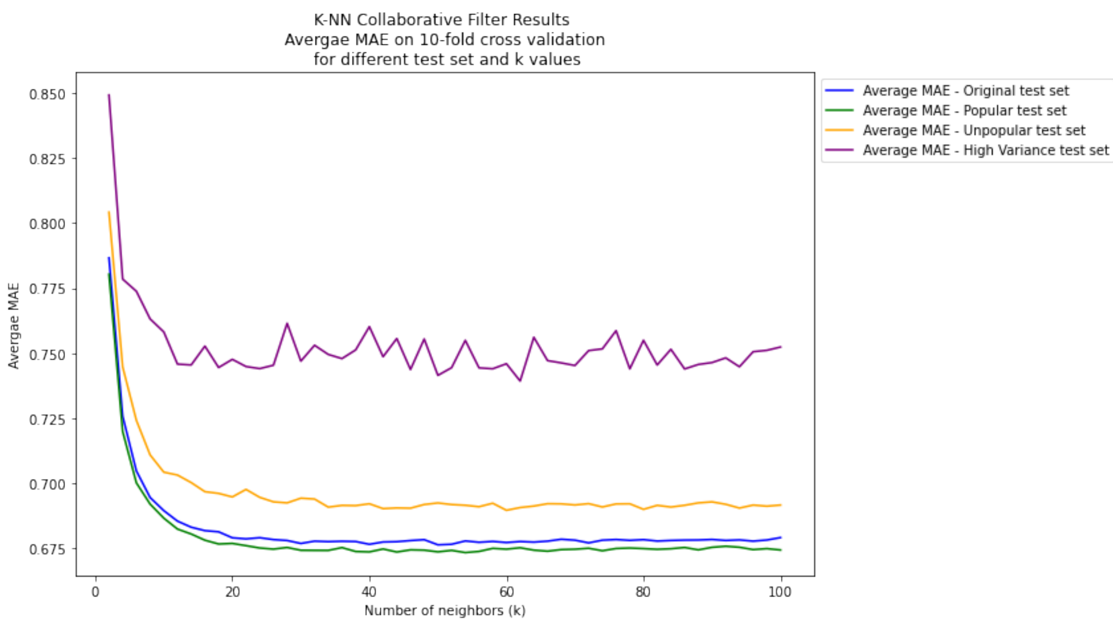
```
Number of popular movies: 4980
Number of unpopular movies: 4744
Number of high variance movies: 87
```

Methods

Results of KNN on Original Test set and Popular, Unpopular, High Variance Trimmed Test Sets For each trimming option I created test sets and repeated the a similar experiment as in question 4. For different neighbor numbers, I trained a KNNwithMeans model, and then evaluated the results in popular, unpopular and high variance test sets for 10-fold, averaged the RMSE results of 10-fold validation.

To be able to compare I also added the original test set in the plots. Original test set in this report refers to as no trimming applied to the test set. I also added the MAE scores out of curiosity.





k	avg_mae_high_variance	avg_mae_original	avg_mae_popular	\
2	0.849334	0.786651	0.780330	
4	0.778469	0.725708	0.719788	
6	0.773789	0.704645	0.700055	
8	0.763165	0.694420	0.691886	
10	0.758136	0.689355	0.686463	
12	0.745751	0.685317	0.682259	
14	0.745432	0.682974	0.680355	
16	0.752689	0.681653	0.677975	
18	0.744441	0.681203	0.676511	
20	0.747583	0.678903	0.676736	
22	0.744823	0.678490	0.675874	
24	0.744016	0.678930	0.674972	
26	0.745330	0.678211	0.674548	
28	0.761454	0.677856	0.675173	
30	0.746899	0.676724	0.674102	
32	0.753008	0.677604	0.674062	
34	0.749480	0.677422	0.674048	
36	0.747882	0.677567	0.675127	
38	0.751217	0.677462	0.673628	
40	0.760170	0.676396	0.673472	
42	0.748603	0.677288	0.674594	
44	0.755604	0.677406	0.673429	
46	0.743674	0.677833	0.674259	
48	0.755417	0.678177	0.674143	
50	0.741425	0.676198	0.673492	
52	0.744389	0.676405	0.674055	
54	0.754895	0.677689	0.673206	
56	0.744301	0.677176	0.673656	
58	0.743961	0.677501	0.674829	
60	0.745902	0.677084	0.674537	
62	0.739239	0.677460	0.675055	
64	0.756075	0.677241	0.674147	
66	0.747105	0.677606	0.673744	
68	0.746221	0.678341	0.674361	
70	0.745223	0.677986	0.674512	
72	0.750958	0.676945	0.674874	
74	0.751619	0.677983	0.673913	
76	0.758635	0.678236	0.674780	
78	0.743913	0.677914	0.674953	
80	0.754924	0.678190	0.674743	
82	0.745413	0.677651	0.674479	
84	0.751456	0.677875	0.674691	
86	0.743885	0.678007	0.675167	
88	0.745621	0.678056	0.674261	

90	0.746350	0.678255	0.675234
92	0.748157	0.677875	0.675651
94	0.744738	0.678085	0.675258
96	0.750525	0.677615	0.674361
98	0.751062	0.678046	0.674735
100	0.752332	0.678980	0.674207

	avg_mae_unpopular	avg_rmse_high_variance	avg_rmse_original \
k			
2	0.804224	1.067826	1.019113
4	0.744675	0.988268	0.944187
6	0.724103	0.973215	0.920493
8	0.710823	0.968007	0.907706
10	0.704164	0.965289	0.902073
12	0.703050	0.945798	0.897826
14	0.700153	0.948333	0.895068
16	0.696647	0.958898	0.893794
18	0.695994	0.945616	0.893812
20	0.694625	0.951085	0.891884
22	0.697524	0.949105	0.890808
24	0.694471	0.948070	0.891313
26	0.692740	0.946224	0.889926
28	0.692344	0.965575	0.890507
30	0.694146	0.948489	0.888361
32	0.693821	0.959835	0.889932
34	0.690703	0.954097	0.890425
36	0.691374	0.948619	0.890208
38	0.691290	0.953159	0.889913
40	0.692000	0.965518	0.888623
42	0.690132	0.953565	0.889337
44	0.690396	0.959140	0.889812
46	0.690297	0.945907	0.890406
48	0.691714	0.960577	0.890190
50	0.692309	0.946601	0.888374
52	0.691701	0.944038	0.887919
54	0.691424	0.957238	0.890168
56	0.690861	0.945291	0.889474
58	0.692205	0.945370	0.889811
60	0.689486	0.952109	0.889560
62	0.690539	0.943163	0.889882
64	0.691117	0.954999	0.889345
66	0.692024	0.949751	0.889778
68	0.691966	0.948236	0.890870
70	0.691544	0.948599	0.889843
72	0.692047	0.956078	0.889235
74	0.690759	0.953125	0.890479
76	0.691938	0.965101	0.890356
78	0.691987	0.947834	0.890124

80	0.689857	0.963460	0.890440
82	0.691387	0.946842	0.889772
84	0.690735	0.955535	0.890124
86	0.691412	0.945390	0.889980
88	0.692337	0.950473	0.890802
90	0.692730	0.946001	0.891320
92	0.691798	0.951957	0.890688
94	0.690343	0.946360	0.890343
96	0.691498	0.954699	0.889925
98	0.691070	0.954109	0.890391
100	0.691504	0.954344	0.891376

	avg_rmse_popular	avg_rmse_unpopular
k		
2	1.012025	1.043699
4	0.937513	0.964471
6	0.914251	0.941319
8	0.904517	0.925878
10	0.899090	0.916294
12	0.894258	0.917615
14	0.891897	0.914878
16	0.889224	0.910009
18	0.888245	0.909444
20	0.888322	0.907289
22	0.887748	0.912324
24	0.886617	0.906864
26	0.886685	0.903328
28	0.886422	0.904601
30	0.885011	0.905621
32	0.886263	0.906945
34	0.885466	0.902974
36	0.887516	0.904518
38	0.884985	0.903591
40	0.884798	0.904831
42	0.886391	0.900764
44	0.885144	0.903144
46	0.885586	0.903547
48	0.885258	0.903835
50	0.885359	0.904768
52	0.886210	0.904741
54	0.884409	0.903233
56	0.885246	0.903244
58	0.886699	0.904557
60	0.886029	0.901251
62	0.886747	0.903457
64	0.885264	0.903462
66	0.885147	0.904022
68	0.886078	0.904595

```
70          0.885769          0.903183
72          0.886792          0.905840
74          0.885538          0.903276
76          0.886173          0.904738
78          0.887258          0.905562
80          0.886529          0.901093
82          0.886193          0.905314
84          0.886330          0.902513
86          0.886759          0.904241
88          0.886508          0.905413
90          0.887697          0.905030
92          0.887401          0.905464
94          0.887369          0.902905
96          0.886680          0.904979
98          0.886278          0.903811
100         0.885365          0.903884
       avg_mae_high_variance  avg_mae_original  avg_mae_popular  \
count              50.000000         50.000000        50.000000
mean                0.752294          0.682372         0.679014
std                 0.015929          0.017116         0.016575
min                 0.739239          0.676198         0.673206
25%                 0.745249          0.677472         0.674144
50%                 0.748019          0.677948         0.674713
75%                 0.754424          0.678453         0.675252
max                 0.849334          0.786651         0.780330

       avg_mae_unpopular  avg_rmse_high_variance  avg_rmse_original  \
count          50.000000               50.000000          50.000000
mean            0.696960                0.955938           0.895322
std             0.018062                0.018347           0.020062
min             0.689486                0.943163           0.887919
25%             0.691160                0.947090           0.889820
50%             0.691952                0.952033           0.890276
75%             0.694065                0.958483           0.891202
max             0.804224                1.067826           1.019113

       avg_rmse_popular  avg_rmse_unpopular
count         50.000000           50.000000
mean           0.891301            0.910447
std            0.019481            0.021964
min            0.884409            0.900764
25%            0.885550            0.903458
50%            0.886465            0.904670
75%            0.887487            0.906608
max            1.012025            1.043699

Min Avg RMSE and Avg MAE for each test set:
avg_mae_high_variance     0.739239
```

```
avg_mae_original          0.676198
avg_mae_popular           0.673206
avg_mae_unpopular         0.689486
avg_rmse_high_variance    0.943163
avg_rmse_original         0.887919
avg_rmse_popular          0.884409
avg_rmse_unpopular        0.900764
dtype: float64
```

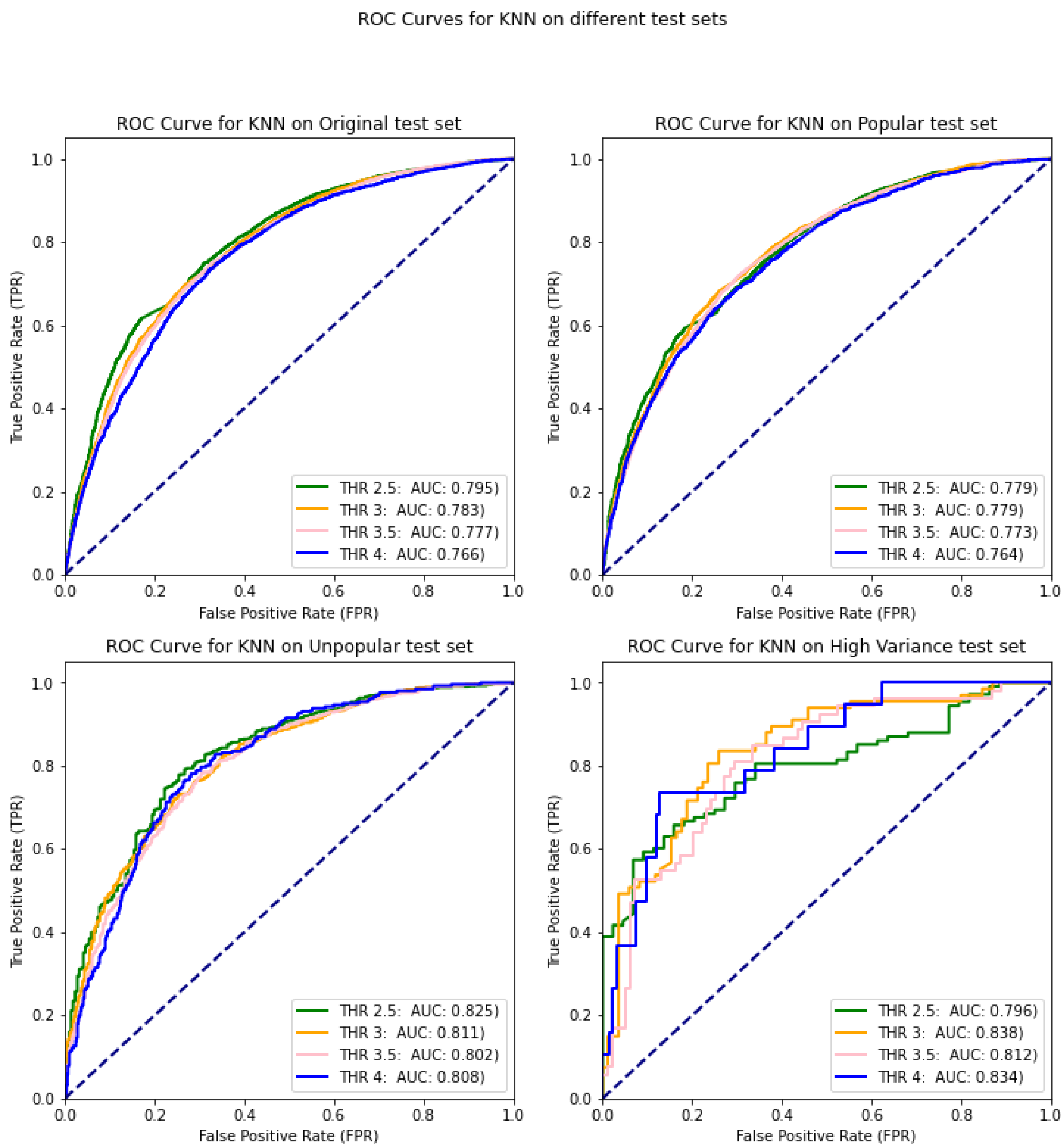
The minimum average RMSE for each test set:

- Original test min AVG RMSE: 0.887919
- Popular test set min AVG RMSE: 0.884409
- Unpopular test set min AVG RMSE: 0.900764
- High variance test set min AVG RMSE: 0.943163

When we look at the RMSE plots, we see that the best RMSE curve belongs to popular test set, even better than original one. This makes perfect sense, since populat test set only contains the movies that have ratings > 2. This means that models, can learn from at least 3 ratings and give good or at least closer predictions to the true rating hence lower the errors. The second best is original test set, which is everything included and unpopular and high variance test sets perform worse. This is again makes sense: for unpopular movie, the number of ratings movies received doesn't really help to predict ratings, and since it is very sparse may be causing overfitting and for the high variance case, this time the problem is that movies got very different ratings, and the big differences between the ratings, causes more trouble and confuses the model. The curve shapes has decreasing trend for original, popular and unpopular ones and reaches to plataeu. For unpopular one however curve is less steady compared the other two, and still has some changes. High variance curve on the other hand, is basically follows a zigzag trend, where the error rates ups and downs very quickly with the change of k. The min AVG. RMSE ratings for each model also shows that high variance set performs the worst.

KNN ROC Curves for each test set For 4 different threshold values [2.5, 3, 3.5, 4], I plotted the ROC curve of each test set. The model used in KNNwithMeans and the number of neighbors (k) is the chosen value in question 5, k=22.

I also added the results for original test set as a reference point.



The Area Under Curve results for each curve in each plot is stated in the legend of the plots.

From the results, for popular test set the threshold with best AUC is threshold=2.5 and 3 with AUC:0.779. The ROC curves and AUC scores for each tresholds for popular movies set are very close to each other. The threshold doesn't seem to make too much difference. For unpopular set, the AUC results are higher, around 0.8-0.82 range. Threshold 2.5 has the highes AUC 0.825, the curves are again similar, though less smoother compared the curves on popular set. For High variance test set, the ROC curves are kind of rectangular shape. The reason for this could be because there is a very low number of samples, 87, in the high variance set (popular and unpopular has more than 4.5K+). Because we are having very few samples for high variance set, the results might not be very reliable. The AUC range is from 0.79-0.838, best threshold is 3 with AUC 0.838.

1.3 Model-based Collaborative Filtering

1.3.1 Question 7

Understanding the NMF cost function: 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 is not convex 1. But it is rather a biconvex optimization problem, this means if we keep U fixed, than the problem is convex for V and we can minimize for V and if we keep V fixed, problem is convex again for U and we can minimize for U. Alternating Least Squares method helps us to iterate between keeping U fixed and solve for V and keeping V and solving for you, until the convergence reached.

For U fixed:

Least-square problem in Equation 5 becomes:

$$\min_{V} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 \tag{5}$$

$$\text{subject to } U, V \geq 0 \tag{6}$$

Equation 7 becomes:

$$\min_{V} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}(r_{ij} - (UV^T)_{ij})^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2 \tag{7}$$

$$\text{subject to } U, V \geq 0 \tag{8}$$

We can show non-covexity of the optimization problem by a counter example:

Assume scalar case where m=n=k=1 and W, r, U and V all scalars. Let's assume also W=1, then the problem in equation 5 becomes

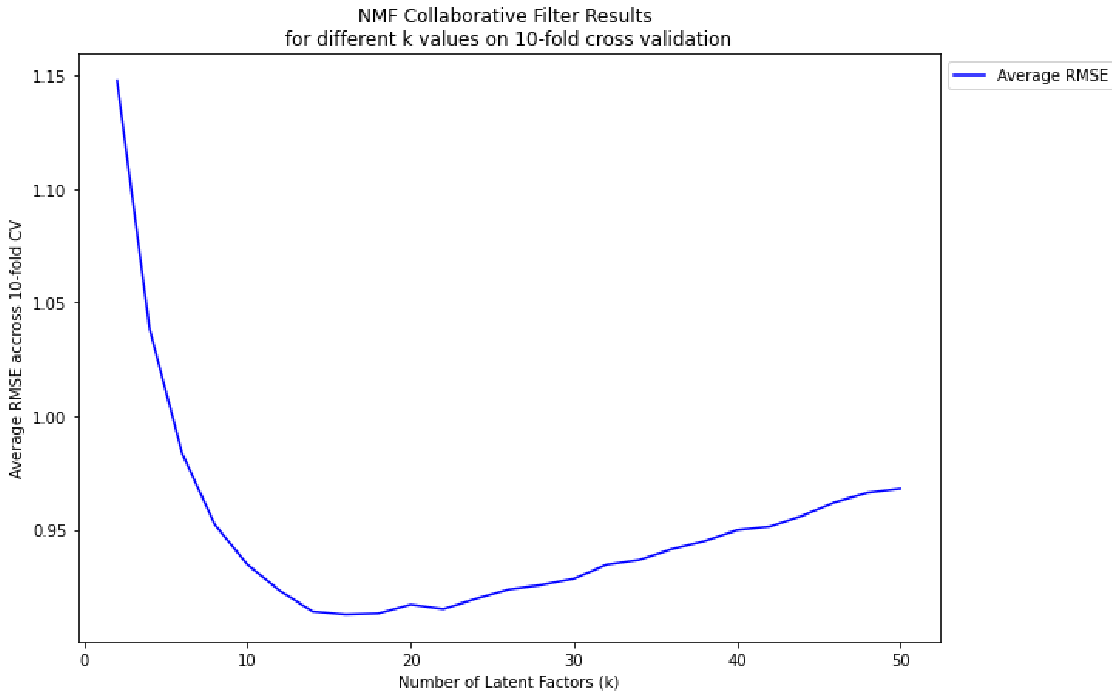
$$\min_{u,v \geq 0} (r - uv)^2 \tag{9}$$

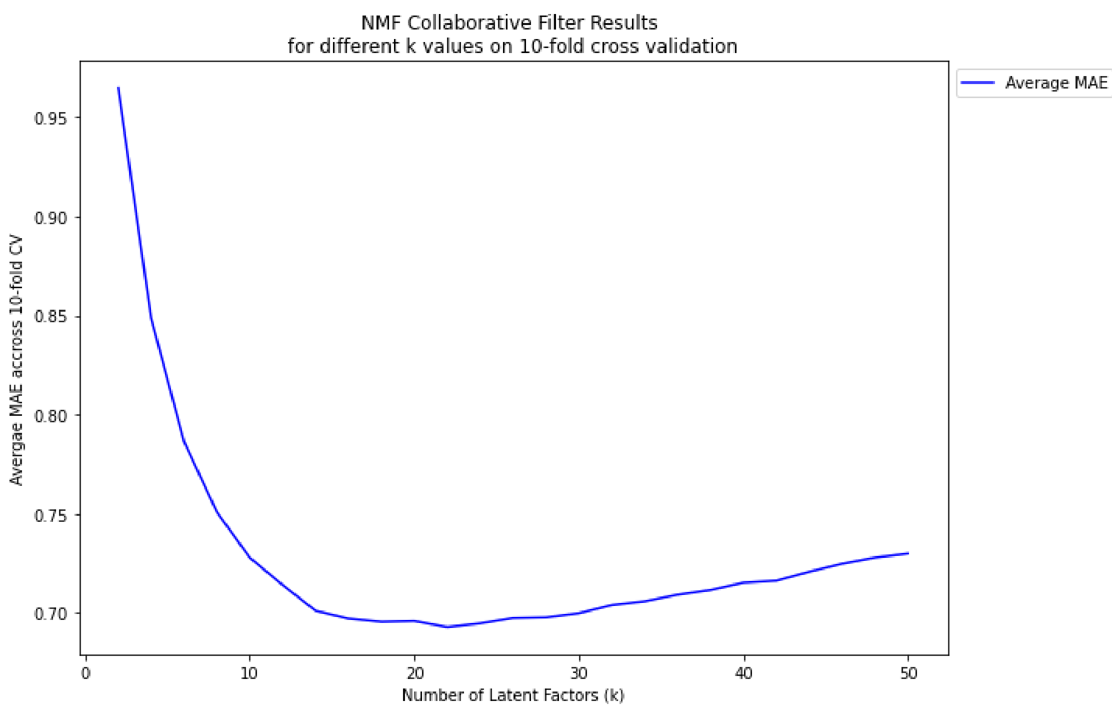
. By taking the Hessian of the equation we want to minimize, and using u=2, v=1 values, we can show that the hessian is not positive semidefinite for all u,v,r >= 0 and hence not convex.

1.3.2 Question 8

Question 8.A Design NMF- Based Collaborative Filter Design a NMF-based collaborative filter to predict the ratings of the movies in the original 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. If NMF takes too long, you can increase the step size. Increasing it too much will result in poorer granularity in your results. 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.

I used NMF model from surprise library to for the NMF collaborative filter. The plots below shows that Average RMSE and MAE results accross 10-fold for each k from 2 to 50, with step sizes 2.





```
                                                          0  \
k                                                         2
avg_rmse                                            1.14788
avg_mae                                            0.964758
cv_results  {'test_rmse': [1.1582446282333843, 1.148348636…

                                                          1  \
k                                                         4
avg_rmse                                            1.03832
avg_mae                                            0.848743
cv_results  {'test_rmse': [1.0388716477142774, 1.035961110…

                                                          2  \
k                                                         6
avg_rmse                                            0.98351
avg_mae                                            0.786312
cv_results  {'test_rmse': [0.9914079911858433, 0.989344772…

                                                          3  \
k                                                         8
avg_rmse                                           0.952192
avg_mae                                            0.750722
cv_results  {'test_rmse': [0.947944592380377, 0.9382680841…

                                                          4  \
k                                                        10
avg_rmse                                           0.934551
avg_mae                                            0.727739
cv_results  {'test_rmse': [0.9274059360624644, 0.936343560…

                                                          5  \
k                                                        12
avg_rmse                                           0.922999
avg_mae                                            0.713942
cv_results  {'test_rmse': [0.9271611637207541, 0.918365858…

                                                          6  \
k                                                        14
avg_rmse                                           0.913905
avg_mae                                            0.700955
cv_results  {'test_rmse': [0.8984714837001442, 0.915702042…

                                                          7  \
k                                                        16
avg_rmse                                           0.912593
avg_mae                                            0.697028
cv_results  {'test_rmse': [0.9145143320746897, 0.913201285…
```

```
                                                         8  \
k                                                       18
avg_rmse                                          0.913035
avg_mae                                           0.695412
cv_results  {'test_rmse': [0.9109177142377667, 0.899319397…

                                                         9  \
k                                                       20
avg_rmse                                          0.917021
avg_mae                                           0.695784
cv_results  {'test_rmse': [0.9238080510847194, 0.926809043…

                                                        10  \
k                                                       22
avg_rmse                                          0.915043
avg_mae                                           0.692744
cv_results  {'test_rmse': [0.9192749622044434, 0.904355120…

                                                        11  \
k                                                       24
avg_rmse                                          0.919551
avg_mae                                           0.694595
cv_results  {'test_rmse': [0.9314006561385201, 0.914919993…

                                                        12  \
k                                                       26
avg_rmse                                          0.923659
avg_mae                                           0.697309
cv_results  {'test_rmse': [0.9460442651764495, 0.922322856…

                                                        13  \
k                                                       28
avg_rmse                                           0.92563
avg_mae                                           0.697638
cv_results  {'test_rmse': [0.9245103333272665, 0.944730195…

                                                        14  \
k                                                       30
avg_rmse                                          0.928382
avg_mae                                           0.699715
cv_results  {'test_rmse': [0.9134075627315653, 0.933350827…

                                                        15  \
k                                                       32
avg_rmse                                          0.934531
avg_mae                                           0.703823
```

```
cv_results  {'test_rmse': [0.948791476853621, 0.9433635199…

                                                          16  \
k                                                         34
avg_rmse                                             0.93666
avg_mae                                             0.705695
cv_results  {'test_rmse': [0.9382978335686227, 0.929346941…

                                                          17  \
k                                                         36
avg_rmse                                            0.941402
avg_mae                                             0.709069
cv_results  {'test_rmse': [0.9313377266396836, 0.927886527…

                                                          18  \
k                                                         38
avg_rmse                                            0.944926
avg_mae                                             0.711432
cv_results  {'test_rmse': [0.9536855353334757, 0.944631422…

                                                          19
k                                                         40
avg_rmse                                             0.94981
avg_mae                                             0.715143
cv_results  {'test_rmse': [0.9468920277983701, 0.956834960…
```

For NMF, we see that both MAE and RMSE curves follow similar trends, at first starts decreasing with increasing k, then starts to increase again after certain number of latent factors. The minimum error rates are when k is in 15-20 range.

Question 8.B Use the plot from the previous part 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?

Min Avg RMSE and MAE values for each k:

	k	avg_rmse	avg_mae
0	2	1.147880	0.964758
1	4	1.038320	0.848743
2	6	0.983510	0.786312
3	8	0.952192	0.750722
4	10	0.934551	0.727739
5	12	0.922999	0.713942
6	14	0.913905	0.700955
7	16	0.912593	0.697028
8	18	0.913035	0.695412
9	20	0.917021	0.695784

10	22	0.915043	0.692744
11	24	0.919551	0.694595
12	26	0.923659	0.697309
13	28	0.925630	0.697638
14	30	0.928382	0.699715
15	32	0.934531	0.703823
16	34	0.936660	0.705695
17	36	0.941402	0.709069
18	38	0.944926	0.711432
19	40	0.949810	0.715143
20	42	0.951334	0.716143
21	44	0.956123	0.720651
22	46	0.962112	0.724703
23	48	0.966273	0.727780
24	50	0.968055	0.729805

Number of Genres in the dataset:

```
20
{'Horror', '(no genres listed)', 'Documentary', 'Film-Noir', 'Thriller',
'Drama', 'Western', 'Fantasy', 'Crime', 'War', 'Adventure', 'Musical',
'Romance', 'Mystery', 'Sci-Fi', 'Animation', 'IMAX', 'Action', 'Comedy',
'Children'}
```

From the Q8-A plot and the values in the above table, we can see that the minimum errors happen:

- min Avg. RMSE = 0.912593 at k=16
- min Avg. MAE = 0.692744 at k=22

The optimal k value should be around 16-22 range. If we compare k=16 and k=22, k=22 seems to be a better choice since MAE is the lowest, and the difference of RMSE value at k=22 and minimum at k=16 is lower compared to choosing k=16 and having min RMSE and a higher difference between MAE scores between two k values. On the other hand, if we care RMSE error more than MAE, k=16 can be chosen as optimal number of latent factors.

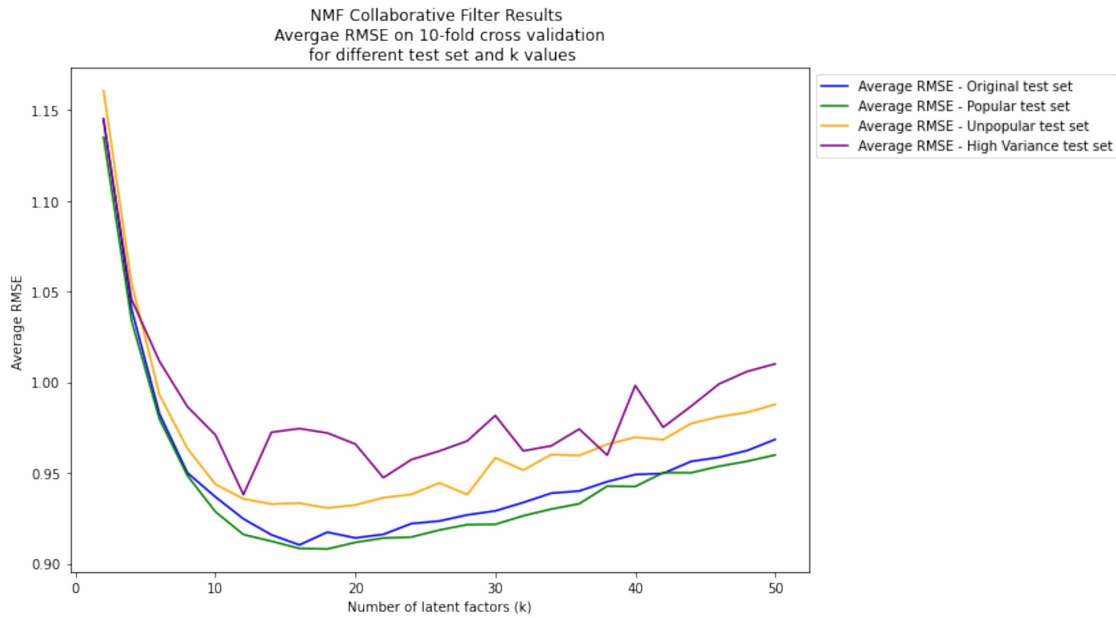
For the following questions we will take k=22, since it has lowest MAE and a close to lowest RMSE score, and proceed with that.

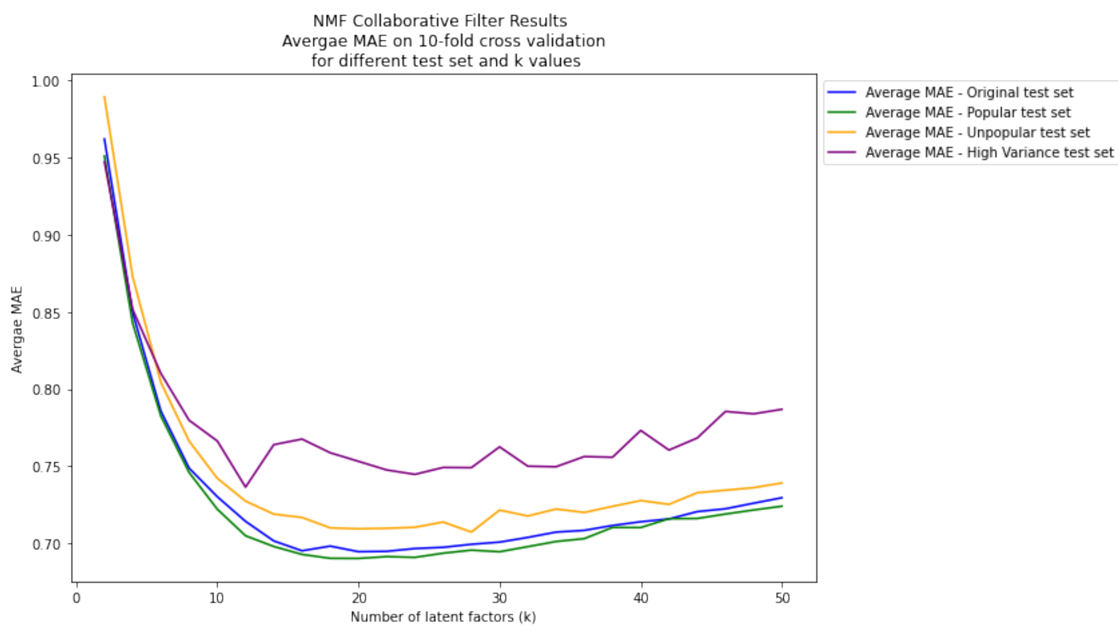
The number of movie genres in the dataset is 20 (counting "no genres listed" as a genre, if not 19). The chosen number of latent factors is close but not same.

Question 8.C Performance on trimmed Test set subsets: For each of Popular, Unpopular and High- Variance test subsets - Design a NMF collaborative filter to predict the ratings of the movies in the trimmed test subset 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); item Report the minimum average RMSE. - Plot the ROC curves for the NMF-based collaborative filter designed in part A for threshold values [2.5, 3, 3.5, 4]. For the ROC plotting use the optimal number of latent factors found in question B. For each of the plots, also report the area under the curve (AUC) value.

Results of NMF on Original Test set and Popular, Unpopular, High Variance Trimmed Test Sets NMF with different values of latent factors fromm 2 to 50 with step size 2 is applied for each trimming type (popular, unpopular and high variance). For different latent factors 10-fold cross validation is applied and avg rmse is created by averaging the RMSE results of 10-fold validations.

To be able to compare I also added the original test set in the plots. Original test set in this report refers to as no trimming applied to the test set. I also added the MAE scores out of curiosity.





k	avg_mae_high_variance	avg_mae_original	avg_mae_popular	\
2	0.947279	0.962131	0.951110	
4	0.851653	0.849588	0.842741	
6	0.810168	0.785723	0.782470	
8	0.779758	0.748722	0.745895	
10	0.766303	0.730156	0.722066	
12	0.736401	0.714199	0.704848	
14	0.763918	0.701502	0.697903	
16	0.767553	0.695109	0.692760	
18	0.758670	0.698148	0.690224	
20	0.753115	0.694522	0.690149	
22	0.747522	0.694757	0.691335	
24	0.744663	0.696569	0.690839	
26	0.749132	0.697358	0.693549	
28	0.749015	0.699315	0.695531	
30	0.762490	0.700725	0.694482	
32	0.749963	0.703780	0.697783	
34	0.749613	0.707228	0.701128	
36	0.756203	0.708344	0.702982	
38	0.755763	0.711509	0.710244	
40	0.773141	0.713939	0.710176	
42	0.760433	0.715762	0.715895	
44	0.768223	0.720501	0.716056	
46	0.785471	0.722339	0.718925	
48	0.783918	0.725963	0.721569	
50	0.786843	0.729507	0.723993	

k	avg_mae_unpopular	avg_rmse_high_variance	avg_rmse_original	\
2	0.989477	1.143929	1.145154	
4	0.872891	1.045634	1.040371	
6	0.804304	1.011401	0.982511	
8	0.766223	0.986427	0.949962	
10	0.742107	0.970860	0.936710	
12	0.727309	0.937976	0.924625	
14	0.718906	0.972358	0.915820	
16	0.716717	0.974399	0.910265	
18	0.709980	0.971958	0.917291	
20	0.709356	0.965859	0.914112	
22	0.709707	0.947326	0.916080	
24	0.710338	0.957279	0.922007	
26	0.713745	0.961990	0.923424	
28	0.707249	0.967574	0.926797	
30	0.721488	0.981581	0.929008	
32	0.717659	0.962083	0.933628	

34	0.722198	0.964864	0.938772
36	0.719960	0.974138	0.939989
38	0.723915	0.959729	0.945133
40	0.727668	0.998117	0.949049
42	0.725214	0.975090	0.949700
44	0.732731	0.986727	0.956298
46	0.734404	0.999007	0.958532
48	0.736040	1.005856	0.962276
50	0.739083	1.010036	0.968389

k	avg_rmse_popular	avg_rmse_unpopular
2	1.135025	1.160615
4	1.033962	1.054709
6	0.979526	0.992989
8	0.948411	0.963353
10	0.928538	0.943703
12	0.915972	0.935643
14	0.912307	0.932804
16	0.908325	0.933305
18	0.908033	0.930619
20	0.911642	0.932290
22	0.914037	0.936286
24	0.914585	0.938120
26	0.918445	0.944404
28	0.921439	0.938046
30	0.921592	0.958296
32	0.926317	0.951502
34	0.930013	0.960029
36	0.932942	0.959558
38	0.942671	0.965734
40	0.942443	0.969618
42	0.950051	0.968288
44	0.950033	0.977222
46	0.953617	0.980914
48	0.956356	0.983345
50	0.959855	0.987695

	avg_mae_high_variance	avg_mae_original	avg_mae_popular	\
count	25.000000	25.000000	25.000000	
mean	0.774288	0.729096	0.724186	
std	0.043327	0.059175	0.058044	
min	0.736401	0.694522	0.690149	
25%	0.749963	0.699315	0.694482	
50%	0.762490	0.711509	0.704848	
75%	0.779758	0.725963	0.721569	
max	0.947279	0.962131	0.951110	

```
       avg_mae_unpopular  avg_rmse_high_variance  avg_rmse_original  \
count          25.000000               25.000000          25.000000
mean            0.743947                0.985288           0.950236
std             0.062308                0.040364           0.049148
min             0.707249                0.937976           0.910265
25%             0.716717                0.964864           0.923424
50%             0.723915                0.974138           0.938772
75%             0.736040                0.998117           0.956298
max             0.989477                1.143929           1.145154

       avg_rmse_popular  avg_rmse_unpopular
count         25.000000           25.000000
mean           0.944646            0.967963
std            0.048360            0.048629
min            0.908033            0.930619
25%            0.915972            0.938046
50%            0.930013            0.959558
75%            0.950051            0.977222
max            1.135025            1.160615
```

```
Min Avg RMSE and Avg MAE for each test set:
avg_mae_high_variance     0.736401
avg_mae_original          0.694522
avg_mae_popular           0.690149
avg_mae_unpopular         0.707249
avg_rmse_high_variance    0.937976
avg_rmse_original         0.910265
avg_rmse_popular          0.908033
avg_rmse_unpopular        0.930619
dtype: float64
```

The minimum average RMSE for each test set:

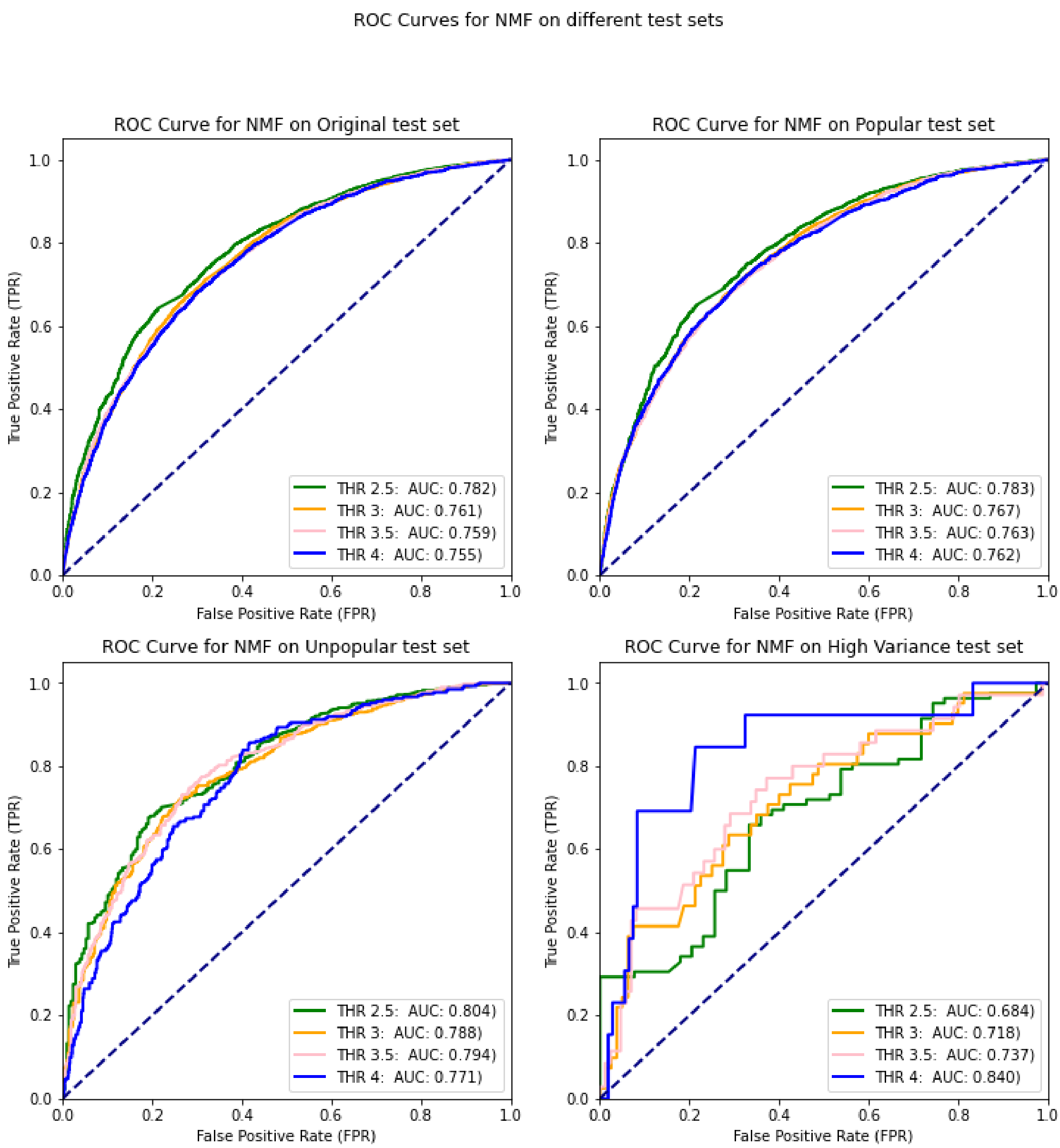
- Original test min AVG RMSE: 0.910265
- Popular test set min AVG RMSE: 0.908033
- Unpopular test set min AVG RMSE: 0.930619
- High variance test set min AVG RMSE: 0.937976

In RMSE plots, the best RMSE curve belongs to popular test set, even better than original one. This makes sense, since popular test set only contains the movies that have ratings > 2, since this test set has lower sparsity compared to others, it helps to have better predictions. The second best is original test set, which is everything included and followed by unpopular and then high variance test sets being the worst one. For unpopular movie, the number of ratings for movies are really sparse, and for the high variance case, movie ratings given were very diverse, and the big differences between the ratings, causes more trouble to the model.

The high level curve trends are similar for all 4 test set: first quick decrease in the error, then slower decrease in the error with higher ks, reaching a minimum error and then starting to have an increasing trend with the increase of k. The smoothness of the curves though is very different. Popular test set curve seems to be more stable and smooth, whereas unpopular one, has small up-down quick changes in the error trend, toward k=30 and the remaining of the experiment. The high variance test set though is very unstable error rates and not smooth after around k=10. trend, then after certain k has decreasing trend for all original, popular and unpopular ones and reaches to plataeu. The min AVG. RMSE ratings for each model also shows that high variance set has the worst min AVG RMSE.

NMF ROC Curves for each test set For 4 different threshold values [2.5, 3, 3.5, 4], I plotted the ROC curve of each test set. The model used in NMF and the number of latent factors (k) is the chosen value in question 8.B, k=22.

I also added the results for original test set as a reference point.



The Area Under Curve results for each curve in each plot is stated in the legend of the plots.

From the results, for popular test set the threshold with best AUC is threshold=2.5 with AUC:0.783. The ROC curves and AUC scores for the rest of the tresholds for popular movies set are around 0.762-0.767 range. The threshold 2.5 AUC and curve seems better but without too much difference. For unpopular set, the AUC results are higher, around 0.771-0.804 range. Threshold 2.5 has the highest AUC 0.804, the curves are not similar, the main difference in the curves happen when the FPR rate is less than 0.4 and TPR less than around 0.8. The curves are also less smooth. For High variance test set, the ROC curves are in staircase shape. The reason for this could be because there is a very low number of samples, 87, in the high variance set (popular and unpopular has more than 4.5K+). Because we are having very few samples for high variance set, the results might not be very reliable. The AUC range is the most different betweeen the other sets, showing thresholding is more important for high variance case compared to the other ones. The AUC range is from 0.684-0.840, best threshold is 3 with AUC 0.84.

1.3.3 Question 9

Interpreting the NMF model: 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?

I performed NMF k=20 for the whole dataset. In surprise library. model.pu is U (user-latent factors) and model.qi is V (movie/item latent factors). For each 20 column of V, I sort the movies by descending order and report the genre of the top 10 movies. Given movies belong to multiple genres, for each column's top 10 movies, I also counted the genre appearance and showed the counts.

V movie-latent factor interaction matrix preview:

```
          0         1         2         3         4         5         6  \
0  0.447124  0.606963  0.278229  0.832767  0.179741  0.022801  0.777665
1  0.430486  0.108222  0.035439  0.205902  0.922796  1.236152  0.530992
2  0.420501  0.030282  0.593468  0.119453  0.136894  0.166774  0.060199
3  0.478775  0.440989  0.523445  0.254507  0.440463  0.539556  0.084055
4  0.415930  0.924954  0.325614  0.487063  0.329309  0.620628  0.576291

          7         8         9  …        11        12        13        14  \
0  0.434236  0.865774  0.371291  …  0.523433  0.934755  0.467175  0.431383
1  0.077147  0.191823  0.453131  …  0.832344  0.060312  0.249052  0.637275
2  0.143417  0.043376  0.261782  …  0.267348  0.374297  0.483580  0.301037
3  0.345769  0.688219  0.289270  …  0.132848  0.659976  0.320904  0.718022
4  0.512560  0.199223  0.034173  …  0.471218  0.562448  0.553170  1.011701

         15        16        17        18        19   movId
0  0.336084  0.594957  0.459337  0.244243  0.390941  112852
1  0.781580  0.395213  0.500689  0.345827  0.551321    1947
2  0.440614  0.160132  0.316761  0.382887  0.360784    1562
3  0.259715  0.647911  0.387552  0.568249  0.729041    2716
```

```
4  0.647172  0.442863  0.427108  0.277184  0.191529   88125

[5 rows x 21 columns]
```

Show first 3 V columns' top 10 movies id, title and genres:

```
---- Column 0 top 10 movies genres: ----

   movieId                                              title  \
0    70946                                   Troll 2 (1990)
1     7116                Diabolique (Les diaboliques) (1955)
2     7564                            Kwaidan (Kaidan) (1964)
3   130634                               Furious 7 (2015)
4    80126                           American, The (2010)
5     2488                             Peeping Tom (1960)
6     3223                     Zed & Two Noughts, A (1985)
7     1173  Cook the Thief His Wife & Her Lover, The (1989)
8     1606                       Kull the Conqueror (1997)
9     4794                                   Opera (1987)

                    genres
0           Fantasy|Horror
1  Horror|Mystery|Thriller
2                   Horror
3    Action|Crime|Thriller
4           Drama|Thriller
5    Drama|Horror|Thriller
6                    Drama
7             Comedy|Drama
8         Action|Adventure
9     Crime|Horror|Mystery

---- Column 1 top 10 movies genres: ----

   movieId                                              title  \
0     5480                             Stuart Little 2 (2002)
1      305                 Ready to Wear (Pret-A-Porter) (1994)
2     2149                     House II: The Second Story (1987)
3    32598                                 Fever Pitch (2005)
4     1376                 Star Trek IV: The Voyage Home (1986)
5    32892  Ivan's Childhood (a.k.a. My Name is Ivan) (Iva…
6      283                          New Jersey Drive (1995)
7   140711                            American Ultra (2015)
8    54734                              Sydney White (2007)
9    34332                                  Sky High (2005)

                         genres
0               Children|Comedy
1                        Comedy
2         Comedy|Fantasy|Horror
```

```
3                  Comedy|Romance
4         Adventure|Comedy|Sci-Fi
5                       Drama|War
6                     Crime|Drama
7     Action|Comedy|Sci-Fi|Thriller
8                          Comedy
9  Action|Adventure|Children|Comedy
```

All V columns' top 10 movies genres:

```
                    V_column_0                        V_column_1  \
movie_0         Fantasy|Horror                   Children|Comedy
movie_1  Horror|Mystery|Thriller                           Comedy
movie_2                 Horror             Comedy|Fantasy|Horror
movie_3    Action|Crime|Thriller                   Comedy|Romance
movie_4         Drama|Thriller           Adventure|Comedy|Sci-Fi
movie_5    Drama|Horror|Thriller                        Drama|War
movie_6                  Drama                      Crime|Drama
movie_7           Comedy|Drama     Action|Comedy|Sci-Fi|Thriller
movie_8       Action|Adventure                           Comedy
movie_9     Crime|Horror|Mystery  Action|Adventure|Children|Comedy

                   V_column_2  \
movie_0    Comedy|Drama|Romance
movie_1            Comedy|Crime
movie_2  Action|Sci-Fi|Thriller
movie_3             Documentary
movie_4             Documentary
movie_5        Romance|Thriller
movie_6         Action|Children
movie_7                  Comedy
movie_8          Comedy|Romance
movie_9               Drama|War

                                                V_column_3  \
movie_0                                      Action|Sci-Fi
movie_1                                             Sci-Fi
movie_2                                      Horror|Sci-Fi
movie_3                                       Comedy|Drama
movie_4                                             Comedy
movie_5                               Action|Drama|Fantasy
movie_6                                      Horror|Sci-Fi
movie_7  Action|Adventure|Animation|Children|Comedy|Rom…
movie_8                                              Drama
movie_9                             Drama|Romance|Western

                                V_column_4  \
movie_0  Action|Fantasy|Horror|Sci-Fi|Thriller
```

```
movie_1                           Comedy|Romance
movie_2                                    Drama
movie_3                  Drama|Horror|Thriller
movie_4                             Action|Drama
movie_5                           Comedy|Fantasy
movie_6                                 Thriller
movie_7                  Horror|Sci-Fi|Thriller
movie_8                       Comedy|Documentary
movie_9                             Comedy|Crime

                                          V_column_5  \
movie_0        Animation|Children|Comedy|Musical
movie_1  Adventure|Animation|Children|Comedy|IMAX
movie_2                           Horror|Thriller
movie_3                            Comedy|Romance
movie_4                 Animation|Children|Comedy
movie_5                                    Horror
movie_6                       Action|Comedy|Crime
movie_7                      Comedy|Drama|Romance
movie_8                             Action|Comedy
movie_9                              Comedy|Drama

                                 V_column_6  \
movie_0  Children|Comedy|Fantasy|Romance
movie_1       Action|Adventure|Animation
movie_2      Action|Comedy|Crime|Fantasy
movie_3                     Comedy|Drama
movie_4            Children|Comedy|Drama
movie_5                           Comedy
movie_6     Action|Comedy|Crime|Thriller
movie_7                            Drama
movie_8                    Action|Comedy
movie_9                   Horror|Mystery

                                        V_column_7  \
movie_0                    Action|Adventure|Sci-Fi
movie_1                              Drama|Romance
movie_2                                Documentary
movie_3                               Action|Drama
movie_4                           Animation|Comedy
movie_5                              Drama|Mystery
movie_6  Comedy|Fantasy|Horror|Musical|Thriller
movie_7                             Comedy|Romance
movie_8                       Comedy|Drama|Romance
movie_9                              Drama|Romance

                                    V_column_8  \
```

```
movie_0              Action|Horror|Thriller
movie_1                     Children|Comedy
movie_2                              Comedy
movie_3                              Comedy
movie_4                              Comedy
movie_5                       Drama|Romance
movie_6  Action|Adventure|Animation|Fantasy
movie_7       Fantasy|Romance|Thriller|IMAX
movie_8                              Horror
movie_9                     Adventure|Drama

                                V_column_9                        V_column_10  \
movie_0                    Fantasy|Western                              Drama
movie_1                     Comedy|Romance                     Drama|Thriller
movie_2       Action|Crime|Sci-Fi|Thriller                     Fantasy|Horror
movie_3                 Adventure|Children             Drama|Romance|Thriller
movie_4  Action|Adventure|Children|Fantasy     Comedy|Crime|Mystery|Romance
movie_5                       Comedy|Drama  Action|Adventure|Fantasy|Sci-Fi
movie_6                             Comedy             Drama|Mystery|Romance
movie_7                Drama|Mystery|Sci-Fi          Action|Adventure|Sci-Fi
movie_8             Children|Comedy|Fantasy                            Horror
movie_9                       Comedy|Drama             Action|Crime|Thriller

                              V_column_11  \
movie_0                   Horror|Thriller
movie_1                  Animation|Comedy
movie_2                      Comedy|Drama
movie_3              Comedy|Drama|Romance
movie_4                   Adventure|Drama
movie_5                         Drama|War
movie_6                      Drama|Horror
movie_7  Action|Adventure|Sci-Fi|Thriller
movie_8                       Crime|Drama
movie_9             Drama|Horror|Thriller

                                  V_column_12  \
movie_0                          Drama|Sci-Fi
movie_1  Crime|Drama|Fantasy|Mystery|Thriller
movie_2                       Horror|Thriller
movie_3                                Comedy
movie_4                                Comedy
movie_5                          Comedy|Drama
movie_6                          Action|Drama
movie_7                                 Drama
movie_8                       Horror|Thriller
movie_9                                Comedy
```

	V_column_13	V_column_14
movie_0	Comedy|Romance	Comedy|Romance
movie_1	Drama|Romance	Drama
movie_2	Comedy|Horror	Crime|Horror|Mystery
movie_3	Drama|Film-Noir	Drama
movie_4	Action|Adventure|Animation|Children|Comedy	Horror|Mystery|Thriller
movie_5	Horror|Thriller	Drama
movie_6	Comedy	Comedy|Drama|Horror
movie_7	Adventure|Thriller	Drama|Thriller
movie_8	Drama|Fantasy|Romance	Comedy|Horror
movie_9	Crime|Drama|Mystery	Drama|Romance

\

	V_column_15	V_column_16
movie_0	Drama|Fantasy|Mystery	Action|Fantasy|Horror|Sci-Fi|Thriller
movie_1	Sci-Fi	Drama
movie_2	Comedy|Drama	Horror
movie_3	Drama|Thriller	Comedy|Drama
movie_4	Drama|Romance	Crime|Drama|War
movie_5	Action|Drama|Fantasy	Animation|Comedy|Drama|Fantasy|Sci-Fi
movie_6	Comedy|Drama|Horror	Drama|Sci-Fi
movie_7	Drama|Mystery|Sci-Fi	Action|Drama
movie_8	Comedy|Romance	Comedy
movie_9	Comedy|Romance	Drama

\

	V_column_17	V_column_18
movie_0	Drama	Drama|Romance
movie_1	Comedy	Comedy|Drama
movie_2	Comedy	Sci-Fi
movie_3	Action|Comedy|Sci-Fi	Drama|Romance|Thriller
movie_4	Adventure|Children|Comedy	Action|Sci-Fi
movie_5	Action|Sci-Fi|Thriller	Action|Comedy
movie_6	Action|Sci-Fi	Drama|Thriller
movie_7	Crime|Drama|Romance	Action|Comedy|Drama
movie_8	Comedy|Musical	Drama
movie_9	Drama|Romance	Comedy|Documentary

\

	V_column_19
movie_0	Action|Comedy|Western
movie_1	Action|Adventure|Drama|Thriller
movie_2	Comedy|Documentary
movie_3	Comedy
movie_4	Comedy
movie_5	Comedy
movie_6	Action|Drama|Sci-Fi
movie_7	Comedy
movie_8	Comedy|Romance
movie_9	Comedy|Fantasy|Romance

Genre counts for top 10 movies for all 20 V columns:

```
Column 0:
        Distinct Genres: 9
        Horror:5 Thriller:4 Drama:4 Mystery:2 Action:2 Crime:2 Fantasy:1
Comedy:1 Adventure:1
Column 1:
        Distinct Genres: 12
        Comedy:8 Children:2 Adventure:2 Sci-Fi:2 Drama:2 Action:2 Fantasy:1
Horror:1 Romance:1 War:1 Crime:1 Thriller:1
Column 2:
        Distinct Genres: 10
        Comedy:4 Romance:3 Drama:2 Action:2 Thriller:2 Documentary:2 Crime:1
Sci-Fi:1 Children:1 War:1
Column 3:
        Distinct Genres: 11
        Sci-Fi:4 Drama:4 Action:3 Comedy:3 Horror:2 Romance:2 Fantasy:1
Adventure:1 Animation:1 Children:1 Western:1
Column 4:
        Distinct Genres: 10
        Thriller:4 Comedy:4 Horror:3 Drama:3 Action:2 Fantasy:2 Sci-Fi:2
Romance:1 Documentary:1 Crime:1
Column 5:
        Distinct Genres: 12
        Comedy:8 Animation:3 Children:3 Horror:2 Romance:2 Action:2 Drama:2
Musical:1 Adventure:1 IMAX:1 Thriller:1 Crime:1
Column 6:
        Distinct Genres: 12
        Comedy:7 Action:4 Drama:3 Children:2 Fantasy:2 Crime:2 Romance:1
Adventure:1 Animation:1 Thriller:1 Horror:1 Mystery:1
Column 7:
        Distinct Genres: 13
        Drama:5 Romance:4 Comedy:4 Action:2 Adventure:1 Sci-Fi:1 Documentary:1
Animation:1 Mystery:1 Fantasy:1 Horror:1 Musical:1 Thriller:1
Column 8:
        Distinct Genres: 11
        Comedy:4 Action:2 Horror:2 Thriller:2 Drama:2 Romance:2 Adventure:2
Fantasy:2 Children:1 Animation:1 IMAX:1
Column 9:
        Distinct Genres: 12
        Comedy:5 Fantasy:3 Children:3 Drama:3 Action:2 Sci-Fi:2 Adventure:2
Western:1 Romance:1 Crime:1 Thriller:1 Mystery:1
Column 10:
        Distinct Genres: 11
        Drama:4 Thriller:3 Romance:3 Action:3 Fantasy:2 Horror:2 Crime:2
Mystery:2 Adventure:2 Sci-Fi:2 Comedy:1
Column 11:
        Distinct Genres: 11
```

```
        Drama:7 Horror:3 Thriller:3 Comedy:3 Adventure:2 Animation:1 Romance:1
War:1 Action:1 Sci-Fi:1 Crime:1
Column 12:
        Distinct Genres: 9
        Drama:5 Comedy:4 Thriller:3 Horror:2 Sci-Fi:1 Crime:1 Fantasy:1
Mystery:1 Action:1
Column 13:
        Distinct Genres: 13
        Comedy:4 Drama:4 Romance:3 Horror:2 Adventure:2 Thriller:2 Film-Noir:1
Action:1 Animation:1 Children:1 Fantasy:1 Crime:1 Mystery:1
Column 14:
        Distinct Genres: 7
        Drama:6 Horror:4 Comedy:3 Romance:2 Mystery:2 Thriller:2 Crime:1
Column 15:
        Distinct Genres: 9
        Drama:7 Comedy:4 Romance:3 Fantasy:2 Mystery:2 Sci-Fi:2 Thriller:1
Action:1 Horror:1
Column 16:
        Distinct Genres: 10
        Drama:7 Sci-Fi:3 Comedy:3 Action:2 Fantasy:2 Horror:2 Thriller:1 Crime:1
War:1 Animation:1
Column 17:
        Distinct Genres: 10
        Comedy:5 Drama:3 Action:3 Sci-Fi:3 Romance:2 Adventure:1 Children:1
Thriller:1 Crime:1 Musical:1
Column 18:
        Distinct Genres: 7
        Drama:6 Comedy:4 Action:3 Romance:2 Sci-Fi:2 Thriller:2 Documentary:1
Column 19:
        Distinct Genres: 10
        Comedy:8 Action:3 Drama:2 Romance:2 Western:1 Adventure:1 Thriller:1
Documentary:1 Sci-Fi:1 Fantasy:1
```

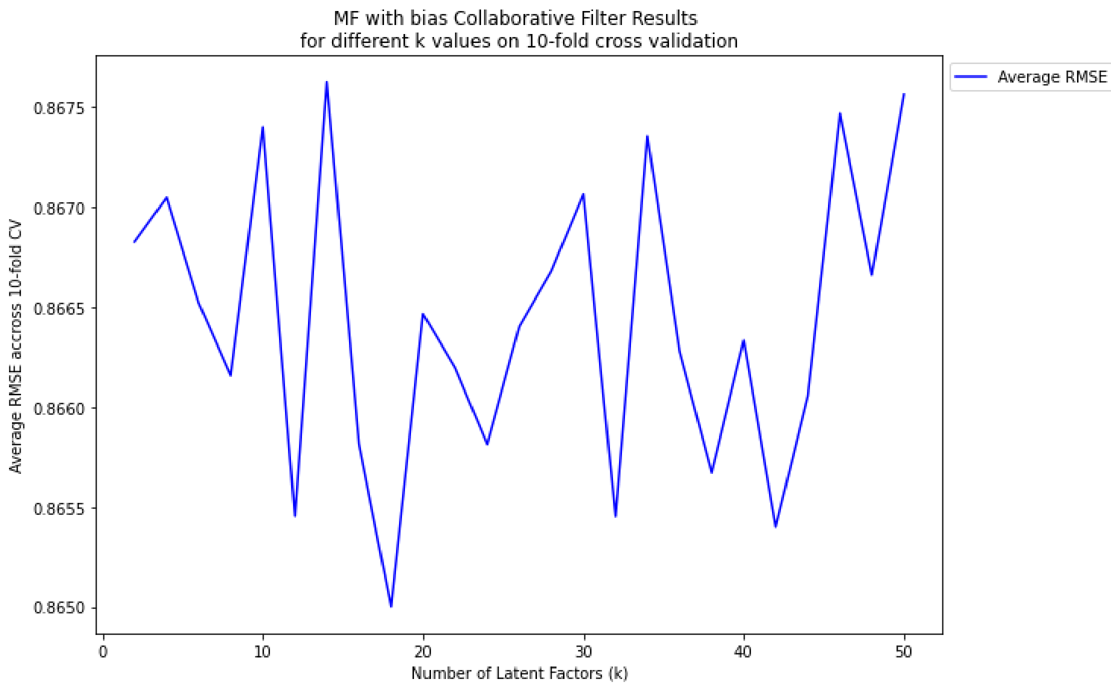
Do the top 10 movies belong to a particular or a small collection of genre? For V columns, latent factors, Top 10 movies doesn't belong to a particular genre but majority indeed belongs to a small collection of genres. For example, in column 0 (first column of V) we can see that the top 10 movies are tagged with 9 distinct genres but the top 3 genres are Horror:5 Thriller:4 Drama:4. We can see that the column 0 latent factor can be represented by a small collection of genres: Horror, Thriller and Drama movies. Similarly column 1 (second column of V) has 9 comedy movie tags out of 10 movies, which can also show second latent factor has a connection with comedy genre. We can see the similar small collection of genres for each latent factor in the above results.

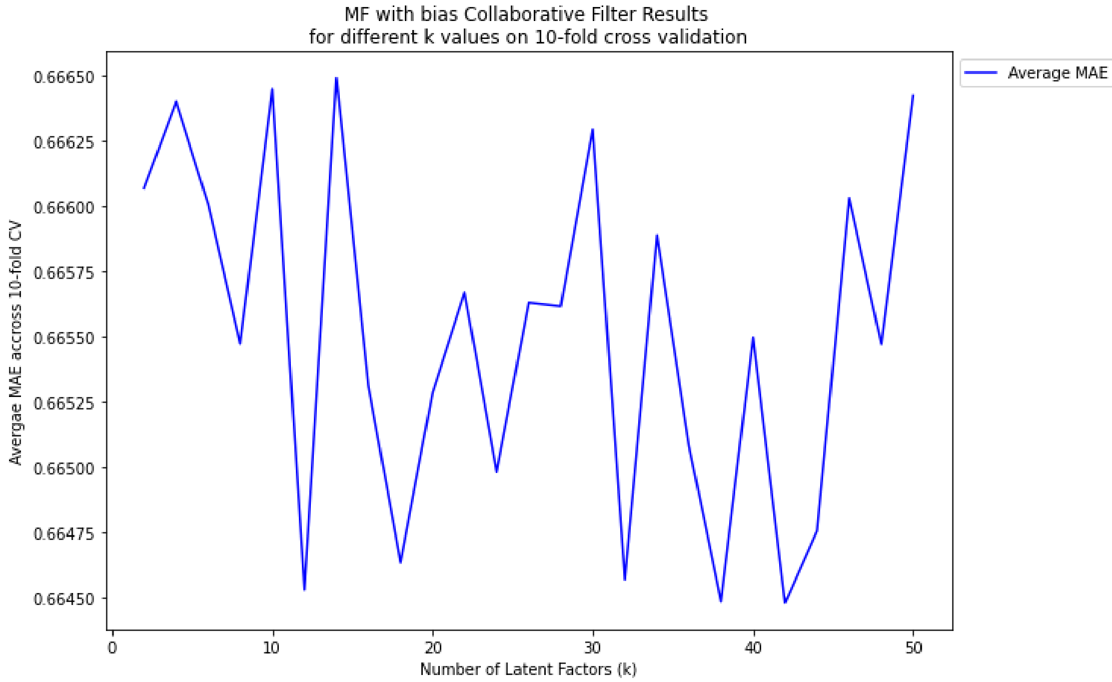
Is there a connection between the latent factors and the movie genres? One of the advantages of NMF is being interpretable, in this specific case, I cannot see a one-to-one mapping with movie genres and latent factors. However, from the above observation, we can say that latent factors have a close connection betwen a small collection of genres. This shows that NMF with the use of latent factors, aggregate movies with similar genres together, and this can improve the recommendations, since this means that model implicitly finds and groups movies with similar collection of genres together and then use latent factors found to do recommendations accordingly.

1.3.4 Question 10

Question 10.A Design MF with bias Collaborative Filter Design a MF-based collaborative filter to predict the ratings of the movies in the original 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.

I used MF with bias=True model from surprise library for the MF with bias collaborative filter. The plots below shows that Average RMSE and MAE results accross 10-fold for each k from 2 to 50, with step sizes 2.






```
                                                          0   \
k                                                         2
avg_rmse                                           0.866823
avg_mae                                            0.666068
cv_results  {'test_rmse': [0.8505212855376368, 0.859773023…

                                                          1   \
k                                                         4
avg_rmse                                           0.867044
avg_mae                                              0.6664
cv_results  {'test_rmse': [0.8566202719465758, 0.884161999…

                                                          2   \
k                                                         6
avg_rmse                                           0.866519
avg_mae                                            0.666005
cv_results  {'test_rmse': [0.869473175027997, 0.8640233258…

                                                          3   \
k                                                         8
avg_rmse                                           0.866155
avg_mae                                            0.665472
cv_results  {'test_rmse': [0.8657014393352603, 0.862854542…
```

```
                                                           4   \
k                                                         10
avg_rmse                                            0.867397
avg_mae                                             0.666448
cv_results  {'test_rmse': [0.8560840212279571, 0.868503524…

                                                           5   \
k                                                         12
avg_rmse                                            0.865453
avg_mae                                              0.66453
cv_results  {'test_rmse': [0.8832030677119755, 0.866466465…

                                                           6   \
k                                                         14
avg_rmse                                            0.867622
avg_mae                                             0.666491
cv_results  {'test_rmse': [0.860445987482174, 0.8637839273…

                                                           7   \
k                                                         16
avg_rmse                                            0.865814
avg_mae                                             0.665311
cv_results  {'test_rmse': [0.858549808253296, 0.8663017184…

                                                           8   \
k                                                         18
avg_rmse                                               0.865
avg_mae                                             0.664633
cv_results  {'test_rmse': [0.8557297531947884, 0.866998337…

                                                           9   …   \
k                                                         20   …
avg_rmse                                            0.866463   …
avg_mae                                             0.665285   …
cv_results  {'test_rmse': [0.866497367725416, 0.8648418851…  …

                                                          15   \
k                                                         32
avg_rmse                                            0.865451
avg_mae                                             0.664568
cv_results  {'test_rmse': [0.8621914707139953, 0.863649863…

                                                          16   \
k                                                         34
avg_rmse                                            0.867351
avg_mae                                             0.665889
cv_results  {'test_rmse': [0.8630598507035923, 0.876562045…
```

```
                                                       17  \
k                                                      36
avg_rmse                                         0.866278
avg_mae                                          0.665083
cv_results  {'test_rmse': [0.864160098123739, 0.8675364609…

                                                       18  \
k                                                      38
avg_rmse                                         0.865669
avg_mae                                          0.664484
cv_results  {'test_rmse': [0.8758539930195344, 0.867151063…

                                                       19  \
k                                                      40
avg_rmse                                          0.86633
avg_mae                                          0.665496
cv_results  {'test_rmse': [0.8667798957130418, 0.850439021…

                                                       20  \
k                                                      42
avg_rmse                                         0.865399
avg_mae                                          0.664479
cv_results  {'test_rmse': [0.8735050572975925, 0.860401127…

                                                       21  \
k                                                      44
avg_rmse                                         0.866054
avg_mae                                          0.664755
cv_results  {'test_rmse': [0.858892899775938, 0.8716084419…

                                                       22  \
k                                                      46
avg_rmse                                         0.867464
avg_mae                                          0.666031
cv_results  {'test_rmse': [0.8789551673500143, 0.865186731…

                                                       23  \
k                                                      48
avg_rmse                                         0.866658
avg_mae                                           0.66547
cv_results  {'test_rmse': [0.8720384805525286, 0.870047159…

                                                       24
k                                                      50
avg_rmse                                         0.867559
avg_mae                                          0.666422
```

```
cv_results  {'test_rmse': [0.8684046427135248, 0.878614088…

[4 rows x 25 columns]
```

For MF with bias, we again see that both MAE and RMSE curves follow similar trends. However, this time the error curves are very different than the previous models we saw. The curve has up and downs and follows a zig zag trend in the error values for different k values. However, if we look closely we can see that the y axis range is very small. The error range for MAE changes between 0.6665-0.6645 which means in the third decimal point and for RMSE curve, the error values are changing between 0.8675-0.8650 which is again in the third decimal point. By taking this into account, we can say that the error ranges are very small, compared to the KNN and NMF models, and MF with Bias also starts with very low error values even in the low k values. It also performs better compared to the other models given that even from start the AVG error metrics were way lower than the min AVG metrics we saw for the previous models.

Question 10.B Use the plot from the previous part 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?

Min Avg RMSE and MAE values for each k:

	k	avg_rmse	avg_mae
0	2	0.866823	0.666068
1	4	0.867044	0.666400
2	6	0.866519	0.666005
3	8	0.866155	0.665472
4	10	0.867397	0.666448
5	12	0.865453	0.664530
6	14	0.867622	0.666491
7	16	0.865814	0.665311
8	18	0.865000	0.664633
9	20	0.866463	0.665285
10	22	0.866194	0.665669
11	24	0.865812	0.664981
12	26	0.866400	0.665629
13	28	0.866676	0.665615
14	30	0.867061	0.666293
15	32	0.865451	0.664568
16	34	0.867351	0.665889
17	36	0.866278	0.665083
18	38	0.865669	0.664484
19	40	0.866330	0.665496
20	42	0.865399	0.664479
21	44	0.866054	0.664755
22	46	0.867464	0.666031
23	48	0.866658	0.665470
24	50	0.867559	0.666422

From the Q10-A plot and the values in the above table, we can see that the minimum errors happen:

- min Avg. RMSE = 0.865000 at k = 18
- min Avg. MAE = 0.664479 at k = 42

The optimal k value chosen is 18, since it falls in the min avg RMSE and its corresponding MAE score is also very close to min AVG MAE. For the following questions I will take chosen k as k=18 for MF with Bias model.

From Question 8.B we know that the number of genres in the dataset is 20 (if we count "no genres listed" as a genre), the chosen number of latent factors is close 18 but not the same.

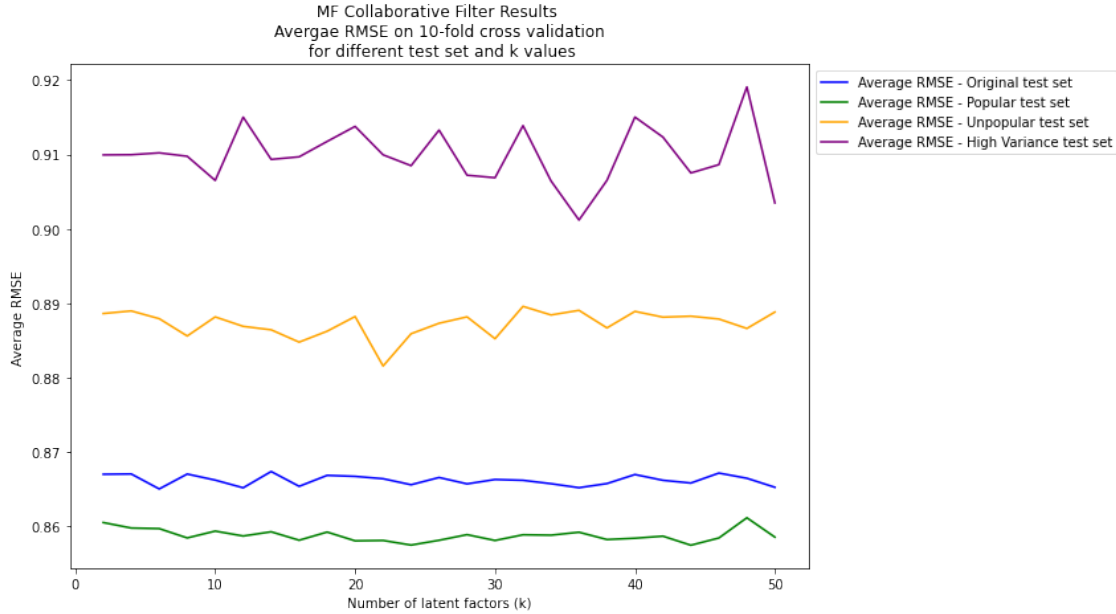
Question 10.C Performance on Test set subsets: For each of Popular, Unpopular and High-Variance test subsets

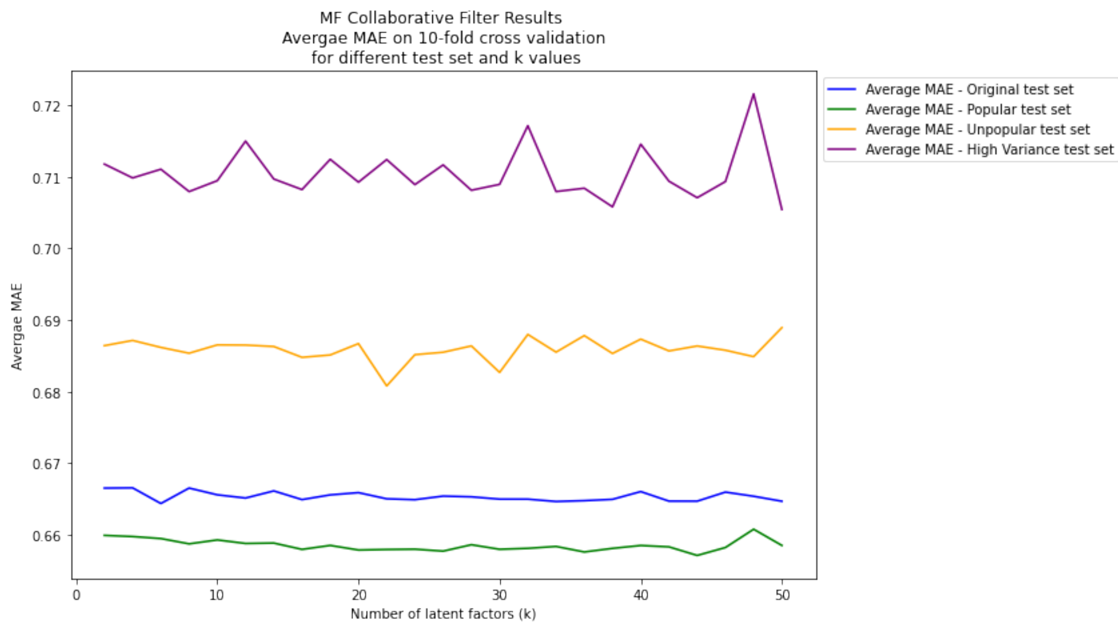
- Design a MF collaborative filter to predict the ratings of the movies in the trimmed test subset and evaluate it's performance using 10-fold cross validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.
- Plot average RMSE (Y-axis) against k (X-axis); item Report the minimum average RMSE.

Plot the ROC curves for the NMF-based collaborative filter designed in part A for threshold values [2.5, 3, 3.5, 4]. For the ROC plotting use the optimal number of latent factors found in question B. For each of the plots, also report the area under the curve (AUC) value.

Results of MF with bias on Original Test set and Popular, Unpopular, High Variance Trimmed Test Sets For MF with bias model, I used different values of latent factors fromm 2 to 50 with step size 2 and applied them for each trimming type (popular, unpopular and high variance). For different latent factors 10-fold cross validation is applied and avg rmse is created by averaging the RMSE results of 10-fold validations.

To be able to compare I also added the original test set in the plots. Original test set in this report refers to as no trimming applied to the test set. I also added the MAE scores out of curiosity.






```
    avg_mae_high_variance  avg_mae_original  avg_mae_popular  \
k
2                0.711792          0.666488         0.659876
4                0.709859          0.666520         0.659712
6                0.711074          0.664332         0.659431
8                0.707949          0.666491         0.658691
10               0.709473          0.665546         0.659237
```

12	0.715006	0.665097	0.658746
14	0.709709	0.666092	0.658801
16	0.708228	0.664878	0.657918
18	0.712462	0.665537	0.658470
20	0.709264	0.665851	0.657829
22	0.712424	0.664988	0.657901
24	0.708924	0.664865	0.657943
26	0.711678	0.665368	0.657672
28	0.708135	0.665266	0.658567
30	0.708958	0.664943	0.657930
32	0.717161	0.664939	0.658065
34	0.707954	0.664614	0.658327
36	0.708418	0.664731	0.657554
38	0.705815	0.664915	0.658059
40	0.714578	0.666000	0.658461
42	0.709397	0.664664	0.658270
44	0.707090	0.664658	0.657070
46	0.709348	0.665928	0.658178
48	0.721616	0.665345	0.660735
50	0.705447	0.664649	0.658470

k	avg_mae_unpopular	avg_rmse_high_variance	avg_rmse_original \
2	0.686406	0.909938	0.866971
4	0.687134	0.909964	0.867009
6	0.686160	0.910214	0.864992
8	0.685348	0.909756	0.867000
10	0.686512	0.906505	0.866183
12	0.686485	0.915025	0.865148
14	0.686286	0.909334	0.867340
16	0.684772	0.909682	0.865345
18	0.685101	0.911751	0.866820
20	0.686689	0.913777	0.866685
22	0.680774	0.909950	0.866370
24	0.685144	0.908488	0.865560
26	0.685483	0.913278	0.866533
28	0.686369	0.907208	0.865676
30	0.682665	0.906868	0.866264
32	0.687974	0.913880	0.866154
34	0.685499	0.906439	0.865700
36	0.687810	0.901176	0.865160
38	0.685311	0.906518	0.865717
40	0.687315	0.915030	0.866934
42	0.685666	0.912335	0.866157
44	0.686357	0.907510	0.865790
46	0.685765	0.908631	0.867131
48	0.684865	0.919087	0.866431
50	0.688922	0.903480	0.865217

k	avg_rmse_popular	avg_rmse_unpopular
2	0.860473	0.888600
4	0.859735	0.888946
6	0.859657	0.887904
8	0.858402	0.885574
10	0.859322	0.888142
12	0.858662	0.886870
14	0.859228	0.886406
16	0.858095	0.884748
18	0.859193	0.886218
20	0.858013	0.888193
22	0.858061	0.881535
24	0.857441	0.885876
26	0.858087	0.887287
28	0.858845	0.888149
30	0.858061	0.885205
32	0.858827	0.889561
34	0.858775	0.888404
36	0.859170	0.889025
38	0.858189	0.886663
40	0.858364	0.888884
42	0.858640	0.888109
44	0.857422	0.888240
46	0.858397	0.887857
48	0.861115	0.886585
50	0.858513	0.888795

	avg_mae_high_variance	avg_mae_original	avg_mae_popular	\
count	25.000000	25.000000	25.000000	
mean	0.710470	0.665308	0.658477	
std	0.003617	0.000650	0.000815	
min	0.705447	0.664332	0.657070	
25%	0.708228	0.664865	0.657930	
50%	0.709397	0.665097	0.658327	
75%	0.711792	0.665851	0.658746	
max	0.721616	0.666520	0.660735	

	avg_mae_unpopular	avg_rmse_high_variance	avg_rmse_original	\
count	25.000000	25.000000	25.000000	
mean	0.685872	0.909833	0.866171	
std	0.001635	0.003911	0.000709	
min	0.680774	0.901176	0.864992	
25%	0.685311	0.907208	0.865676	
50%	0.686160	0.909756	0.866183	
75%	0.686512	0.912335	0.866820	
max	0.688922	0.919087	0.867340	

```
       avg_rmse_popular  avg_rmse_unpopular
count         25.000000           25.000000
mean           0.858747            0.887271
std            0.000864            0.001769
min            0.857422            0.881535
25%            0.858095            0.886406
50%            0.858640            0.887904
75%            0.859193            0.888404
max            0.861115            0.889561
```

```
Min Avg RMSE and Avg MAE for each test set:
avg_mae_high_variance     0.705447
avg_mae_original          0.664332
avg_mae_popular           0.657070
avg_mae_unpopular         0.680774
avg_rmse_high_variance    0.901176
avg_rmse_original         0.864992
avg_rmse_popular          0.857422
avg_rmse_unpopular        0.881535
dtype: float64
```

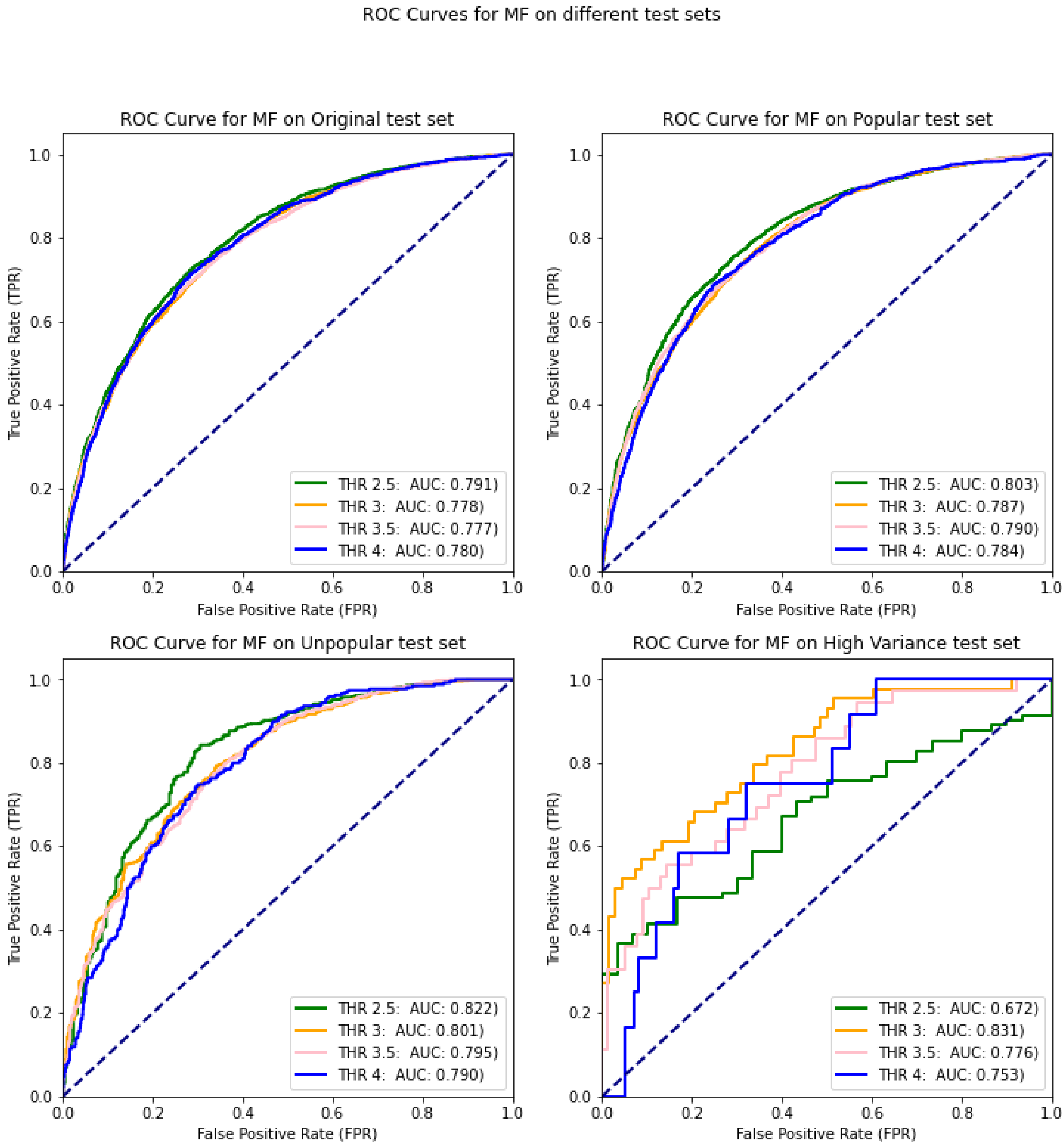
The minimum average RMSE for each test set:

- Original test min AVG RMSE: 0.864992
- Popular test set min AVG RMSE: 0.857422
- Unpopular test set min AVG RMSE: 0.881535
- High variance test set min AVG RMSE: 0.901176

For the test sets we again see the same order popular has best results with lowest min avg RMSE with 0.857422 and overall curve. Followed by original set, unpopular and then high variance set. One difference in the RMSE and MAE plots above from the other models is actually MF with Bias, starts with relatively lower error values and curves have a very small decreasing slope, even like plateau since the error changes mostly happens in 2nd-3rd decimal points. The popular and original plots are more stable where unpopular and high variance sets follow more zigzags during the experiment.

MF with Bias ROC Curves for each test set For 4 different threshold values [2.5, 3, 3.5, 4], I plotted the ROC curve of each test set. The model used in MF with bias and the number of latent factors (k) is the chosen value in question 10.B, k=18.

I also added the results for original test set as a reference point.



The Area Under Curve results for each curve in each plot is stated in the legend of the plots.

From the results, Popular set again has the smoother curves and the difference betweeen the AUC of the thresholds are not too different in the range of 0.784-0.803. This means for popular set any threshold may result in similar results but threshold 2.5 is the best (similar to previous models). Unpopular threshold curve differences are more clear compared to previous models ranging from 0.790 to 0.822. The best threshold for unpopular test set in again 2.5 with AUC 0.822. For high variance set the curves are drastically different and staircase shaped. The best threshold seems like 3 with AUC 0.31 and the worst one is 2.5 with 0.672 AUC. However, as we stated in the previous questions high variance test set has very few samples, 87, which might explain the stair-like curve appearance and the threshold choice and ROC curve might not be very reliable.

1.4 Naive collaborative filtering

1.4.1 Question 11

Designing a Naive Collaborative Filter:

- Design a naive collaborative filter to predict the ratings of the movies in the original 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. Note that in this case, when performing the cross-validation, there is no need to calculate 's i for the training folds each time. You are only asked to use a single set of 's calculated on i the entire dataset and validate on 10 validation folds.
- Performance on Test set subsets: For each of Popular, Unpopular and High-Variance test subsets -
  - 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.

For Naive Collaborative filter, I created a ratings_per_user pandas dataframe shown below, ratings_per_user is raw ratings data grouped by users and rating shows the mean rating for each user calculated on the entire dataset. I also created a get_naive_predictions_from_test function where it takes the ratings_per_user dataframe and test set, it returns the average user rating for each user in the test set. The result of this function is used as the predictions of the test set.

ratings_per_user preview 10 predictions:

calculates mean ratings for each user which will be used as naive collab. filter predictions

```
                    0         1         2         3         4         5  \
userId       1.000000  2.000000  3.000000  4.000000  5.000000  6.000000
mean_rating  4.368534  3.948276  2.435897  3.555556  3.636364  3.493631

                    6         7        8          9
userId       7.000000  8.000000  9.00000  10.000000
mean_rating  3.230263  3.574468  3.26087   3.278571
```

Some stats on the ratings_per_user dataframe

As seen from below the mean prediction is 3.65 where min assigned prediction is 1.275 and max is 5

```
             count        mean         std    min     25%         50%  \
userId       610.0  305.500000  176.236111  1.000  153.25  305.500000
mean_rating  610.0    3.657226    0.480641  1.275    3.36    3.694385

                  75%    max
userId       457.7500  610.0
mean_rating    3.9975    5.0
```

For all 4 test set, original (no trim) , popular, unpopular and high_variance, I performed Naive Colab. Filter with 10-fold cross validation, average RMSE is calculated for each by averaging the

RMSE results across all 10 folds. The results are following:

```
Naive Collaborative Filter on Original Average RMSE: 0.934663052844902
Naive Collaborative Filter on Original Average MAE: 0.7289328699138828
Naive Collaborative Filter on Popular Average RMSE: 0.9250802897934453
Naive Collaborative Filter on Popular Average MAE: 0.7214231849077074
Naive Collaborative Filter on Unpopular Average RMSE: 0.9705068660276013
Naive Collaborative Filter on Unpopular Average MAE: 0.7561299367998391
Naive Collaborative Filter on High_variance Average RMSE: 0.9211233251598134
Naive Collaborative Filter on High_variance Average MAE: 0.7192754474476302
```

Naive Collab Filter:

- Original set Average RMSE: 0.934663052844902
- Popular set Average RMSE: 0.9250802897934453
- Unpopular set Average RMSE: 0.9705068660276013
- High Variance set Average RMSE: 0.9211233251598134

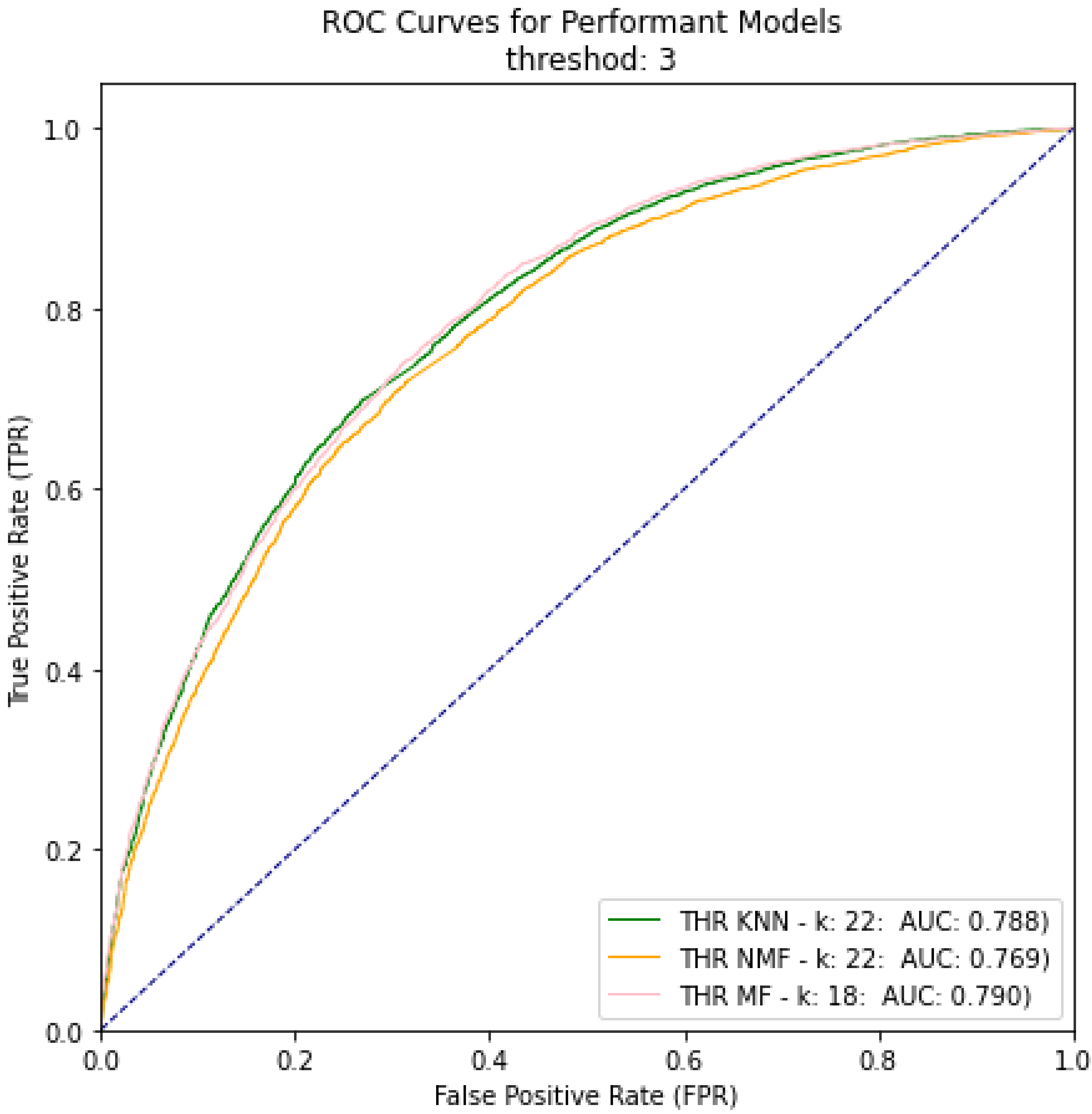
The Average RMSE results are mostly worse than the previous models min AVG RMSE results, but the results are not too different. This shows that even using this simple idea of averaging users previous ratings and used that as a recommended rating can give somewhat acceptable performance. The most interesting thing in this result if the Avg. RMSE of high variance set, in all the previous models, this set was the worst error rates, however in this model it is even better than the original set. My guess is that when the variance is too high between movie ratings, the mean find by the naive algorithm will be in the middle ratings, 3ish, showing moderate interest in the movie rather than strong preference, which will decrease the squared error of RMSE. For example, let's say we have a movie with 0.5 and 5, 2 total ratings, the recommended mean will be 2.75, which is moderate interest, there is no strong preference. So, even if the prediction is wrong the squared difference for different movies, will stay lower, compared than guessing 0.5 for a 5, high true rating.

1.5 Performance Comparison

1.5.1 Question 12

Comparing the most performant models across architecture: Plot the best ROC curves (threshold = 3) for the k-NN, NMF, 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.

I plotted the best ROC curves with threshold 3 for performant models on KNN, NMF and MF with their chosen k values: 22, 22, 18 respectively.



The ROC curves are close together and smooth, we can seee that yellow one, NMF is slightly worse than the other two. MF with bias has the best AUC with 0.79, hence best at predicting movie ratings among the rest of the models, followed by KNN with 0.788, followed by NMF with 0.769. The differences between KNN and MF with bias is very low and either model could be a good choice for movie ratings.

1.6 Ranking

1.6.1 Question 13

Understanding Precision and Recall in the context of Recommender Systems: 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.

Relevant items to the user is where ground truth is positive, which means movies liked by user.

Precision:

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

Precision is the amount of movies where intersection of recommended movies by system and ground truth positives (liked by user) over number of recommended movies. The metrics meaning is: what rate (percentage) of the recommended items is also liked by the user. In another words, it checks how relevant the recommended items to the user by comparing it against the ground truth positives. This metric helps us to understand how good the recommended items.

Recall:

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

Recall is the amount of movies where intersection of recommended movies by system and ground truth positives (liked by user) over grounf truth positives. It tells what rate (percentage) of movies liked by the user is in recommmended items list. In another words, it checks the percentage of relevant items being recommended to the user. Recall helps us understand how good the recommender model in finding/recommending items liked by the user.

1.6.2 Question 14

Understanding Precision and Recall in the context of Recommender Systems: 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.

In this question, I used KNN, NMF and MF with bias models with their chosen ks (22,22,18 respectively) to plot individual and combined Precision vs t, Recall vs t and Precision Recall curves. I thresholded the Ground Truth values with 3, and assigned 1 for the movies higher than 3 and 0 for the ones lower or equal to 3. When creating the ground truth positives (G) list, for test set, I discarded the users whose $|G| = 0$, they didn't like any movie. I also dropped the users who rated less than t movies, for the test set on t.

The experiment is ran t from 1 to 25 by stepsizes 1. I applied 10 cross validation for each t and average the Precision and Recall values. The Precision and Recall is calculated for each t by using precision_recall_at_t_per_user method I wrote. What the method does, given predictions, t value and a threshold, it returns a average user precision and average user recall.

To show better how precision_recall_at_t_per_user do its work, I ran a sample KNN model with k=20, did some predictions on 10% of the data and feed it to the precision_recall_at_t_per_user. Below shows the dataframe created within the function:

rec_order_items: ordered ids of the movies based on given highest predictions for the user (the list of all the movies ordered by recommendation preference of the model) rec_t_items: the first t items in the rec_order_items (given a t, returns t movies with highest predicted ratings for that user) liked_items: G, ground truth positives of the user user_precision_at_t: user precision at t, uses the precision formula in Q13 user_recall_at_t: user recall at t, uses the recall formula in Q13

t value is 5 in below sample run and threshold is 3.

```
   uid                                     rec_order_items  \
0    1  [1136, 101, 3703, 1291, 2395, 1408, 1240, 2657…
1    2         [6874, 112552, 99114, 68157, 80906, 79132]
2    4  [319, 2019, 162, 3358, 1265, 171, 902, 2186, 1…
3    6  [536, 475, 490, 47, 216, 151, 165, 2, 628, 371…
4    7  [1270, 260, 48516, 1220, 1517, 588, 4874, 4927…
5    9                        [187, 922, 1198, 223, 5507]
6   10  [70183, 8970, 73017, 119145, 6942, 86548, 1247…
7   11                 [1408, 153, 1385, 1687, 1586, 511]
8   15  [318, 1200, 2571, 2150, 1198, 1653, 7438, 1270…
9   16  [858, 1267, 260, 7361, 58559, 3741, 4993, 47, …

                           rec_t_items  \
0       [1136, 101, 3703, 1291, 2395]
1  [6874, 112552, 99114, 68157, 80906]
2        [319, 2019, 162, 3358, 1265]
3            [536, 475, 490, 47, 216]
4      [1270, 260, 48516, 1220, 1517]
5         [187, 922, 1198, 223, 5507]
6  [70183, 8970, 73017, 119145, 6942]
7       [1408, 153, 1385, 1687, 1586]
8       [318, 1200, 2571, 2150, 1198]
9       [858, 1267, 260, 7361, 58559]

                                         liked_items  user_precision_at_t  \
0  [349, 3033, 1136, 101, 2985, 2116, 2450, 3703,…                 1.0
1         [99114, 80906, 6874, 112552, 68157, 79132]                 1.0
2  [1219, 176, 1265, 1266, 902, 1449, 2692, 106, …                 0.6
3  [248, 367, 314, 60, 475, 450, 47, 46, 93, 43, …                 0.6
4         [1270, 5445, 49272, 1517, 1220, 260, 2717]                 0.8
5                                  [1198, 223, 922]                 0.6
6           [1784, 49286, 6942, 7458, 7375, 72737]                 0.2
7                                  [511, 1408, 1586]                 0.4
8  [1270, 2012, 296, 2150, 1653, 2571, 85414, 318…                 1.0
9          [3741, 47, 3174, 4993, 1267, 58559, 7361]                 0.6

   user_recall_at_t
0          0.312500
1          0.833333
2          0.150000
3          0.187500
4          0.571429
5          1.000000
6          0.166667
7          0.666667
8          0.416667
9          0.428571
```

```
Avg Precision at t per user: 0.7567307692307691
Avg Recall at t per user: 0.5041367878300609
```

Then the average precision at t for all users is the average of user_precision_at_t and the average recall at t for all users is the average of user_recall_at_t. These two values are returned by method. In the experiment I did for this question, since we do 10-folds cross validation, we repeat the procedure for 10 times for each t and take average of the 10 precision and recall values found for each t and use those results in our plots.

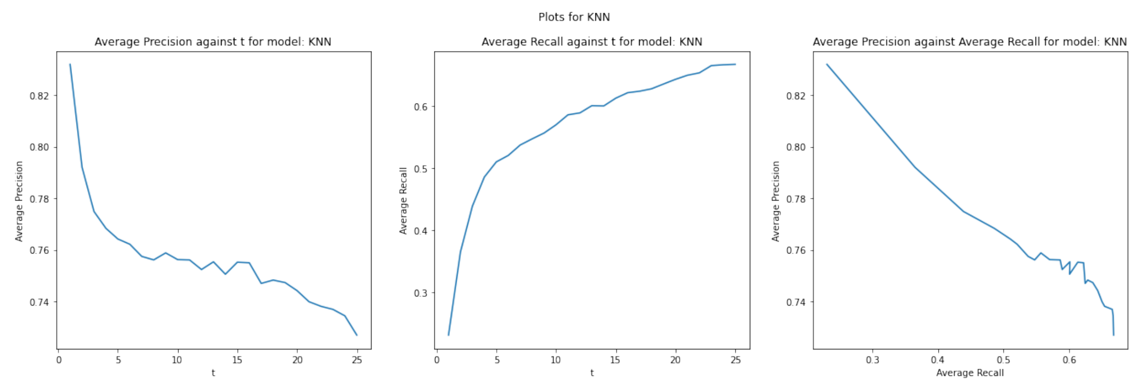
Resulting data frame of the t from 1 to 25, precision-recall experiment:

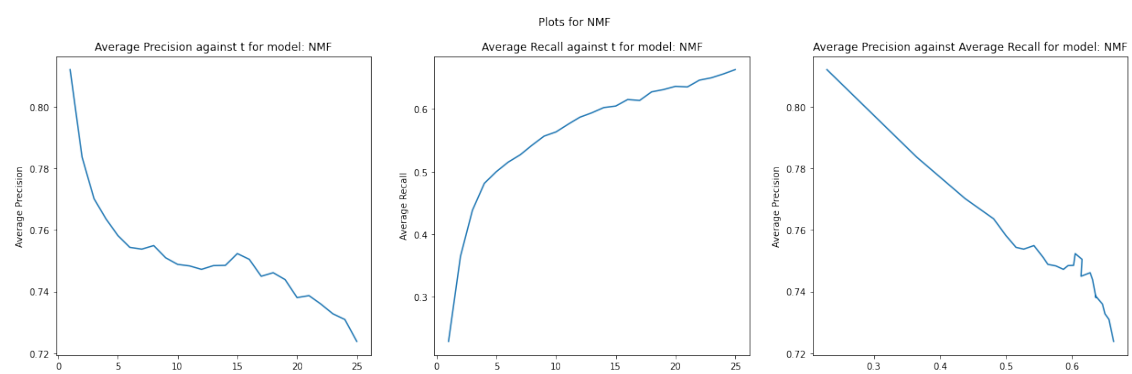
t	avg_precision_KNN	avg_precision_MF	avg_precision_NMF	avg_recall_KNN \
1	0.831868	0.841300	0.812111	0.230972
2	0.791983	0.799967	0.783713	0.365116
3	0.774855	0.786536	0.770202	0.438964
4	0.768327	0.774579	0.763638	0.485876
5	0.764198	0.768870	0.758205	0.510270
6	0.762171	0.766397	0.754364	0.520674
7	0.757455	0.763874	0.753792	0.537494
8	0.756103	0.763794	0.754950	0.547400
9	0.758818	0.761600	0.750980	0.556838
10	0.756222	0.759600	0.748847	0.570113
11	0.756098	0.756379	0.748343	0.586192
12	0.752369	0.754978	0.747223	0.589309
13	0.755360	0.758462	0.748443	0.600971
14	0.750557	0.757257	0.748520	0.600555
15	0.755203	0.756398	0.752352	0.613024
16	0.754985	0.754282	0.750478	0.621862
17	0.746986	0.753730	0.744982	0.624278
18	0.748288	0.750632	0.746139	0.628209
19	0.747310	0.750711	0.743896	0.635878
20	0.744195	0.748451	0.738054	0.643450
21	0.739896	0.749442	0.738697	0.649982
22	0.738127	0.741284	0.735933	0.653876
23	0.736931	0.739676	0.732784	0.665538
24	0.734440	0.735565	0.730945	0.666905
25	0.726969	0.734443	0.723822	0.667597

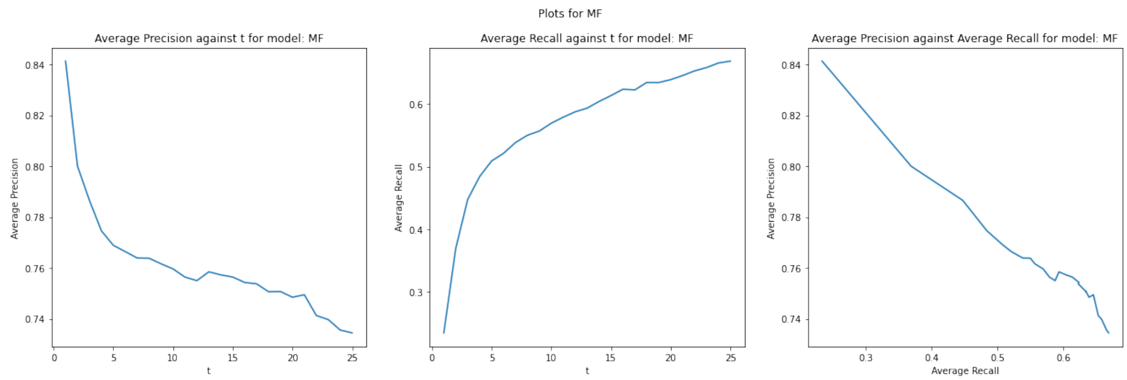
t	avg_recall_MF	avg_recall_NMF
1	0.233830	0.228701
2	0.369130	0.364565
3	0.447517	0.438111
4	0.484154	0.481287
5	0.509054	0.499875
6	0.521323	0.515219
7	0.538607	0.526829

8	0.550214	0.542325
9	0.557049	0.556715
10	0.569562	0.563418
11	0.579185	0.575528
12	0.587800	0.586970
13	0.593623	0.593982
14	0.604341	0.602353
15	0.613780	0.604805
16	0.623956	0.615179
17	0.622911	0.613696
18	0.634852	0.627447
19	0.634722	0.631101
20	0.639297	0.636246
21	0.645882	0.635310
22	0.653296	0.646143
23	0.658665	0.649880
24	0.665920	0.656017
25	0.669018	0.662987

Individual Plots for each model Precision vs t, recall vs t and precision-recall plots for each model drawn separately in below:







Precision vs t plots:

In all precision vs t plots, we can see the non-monotonic decreasing trends. As t increase the precision mostly decrease and this shows an inverse relation between t and average precision. For KNN, there is a smooth decreasee until the middle of t=5 and 10, and then the curve becomes less smooth, during the experiment from start to end precision lowers more than 0.08. For NMF, we again see similar curve but the smoothness is preserved longer, around until t= 12-13. For MF with bias, the precision vs t plot is the smoothest one, and the one that has highest precision range, it starts from 0.84 and drops until below 0.74.

Recall vs t plots:

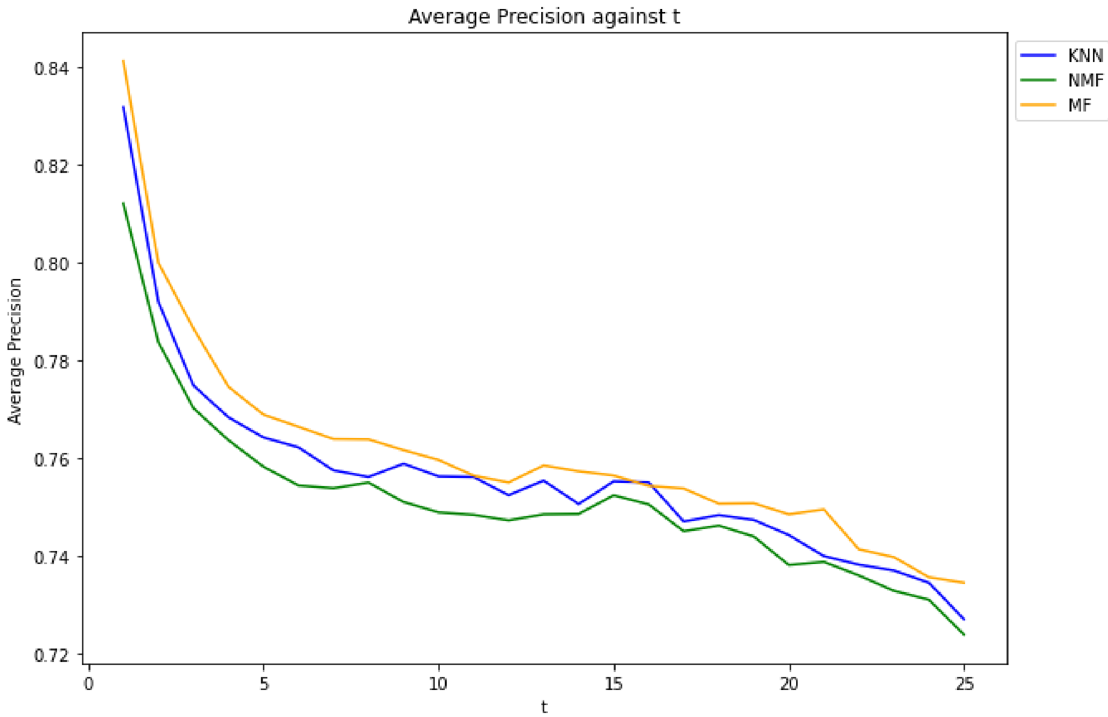
The recall vs t relationship is positive. As t increase recall also increases, which makes sense, the more movie we recommend the more of the liked movies by the user we can cover. The curves for all 3 models follows the same trend.

Avg Precision vs Avg Recall plots:

There is an inverse relationship between precision and recall. As recall increases the precision decrease for all 3 plots. The trends for the curves same for all 3 models. MF with bias has the highest precision rate when the recall is smallest among the models. Also for all 3 plots, the curves become less smooth for the recall values above 0.5. We can also clearly see the precision-recall tradeoff from these plots since these two metrics are in inverse relationship. We need to find an optimal t point where both recall and precision is acceptable.

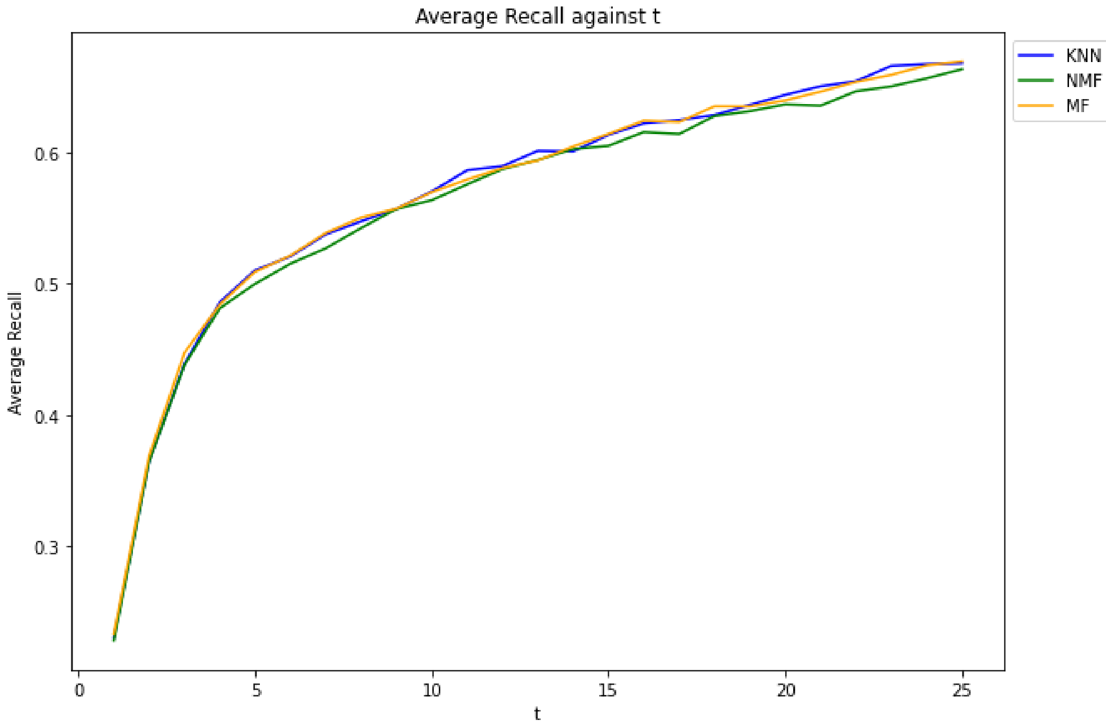
Combined Plots I also plotted the combined plots for each type of plots, to be able to compare the models better:

Average Precision against t



From the above plot, we can see that MF with bias overall has the best and smoothest precision vs t plot. It starts and finish with higher precision values for different ts. If we want to have high precision generally it seems best to keep the t lower. KNN is the second best model in terms of precision and NMF is the worst one.

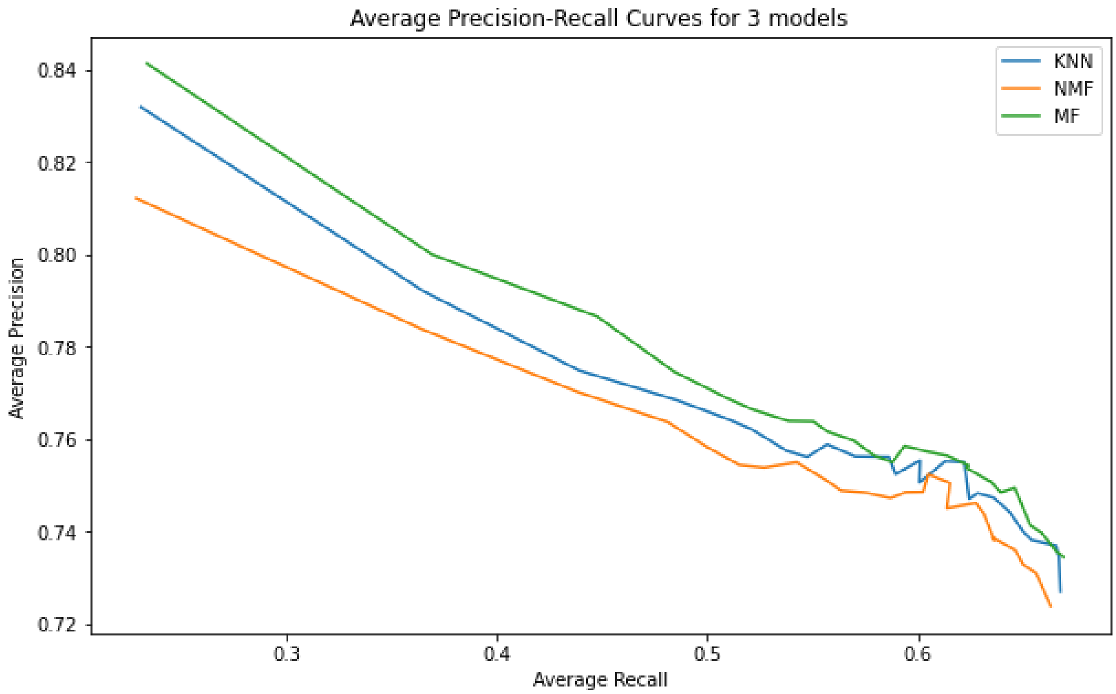
Average Recall against t



The recall curves for each model are more similar to each other compared to precision curves. From the above plot it is very hard to tell which model is better in terms of recall because the marginal differences are very close. NMF looks slighlt worse then the other 2 models though.

Given that we saw precision and recall tradeoff in individual plots, and seeing no high difference in terms of recall in these models, the model selection can be made based on the model which has good precision. In this case, MF with bias would be a better model compared the others for the recommendation system of the given data.

Precision - Recall Curve Combined



All three models has inverse precision-recall relationship as described above plots. From combined recall plots, we saw that there is not much difference of the recall against different t's in the models, hence the main driver and difference we see in this plot is coming from the precision differences of the models. In precision-recall curves the closer the line to the upper right side, the better model for recommendation for this dataset is, in this case the best model for this dataset is MF with bias. Since the slope of the green line is also smaller (closer to the right upper side), we can do the precision-recall trade-off better because increasing recall would decrease the precision less than the other models. So, we can minimize the trade-off by choosing MF with bias as our best model. The second best choice seems to be KNN and the worst one is NMF.