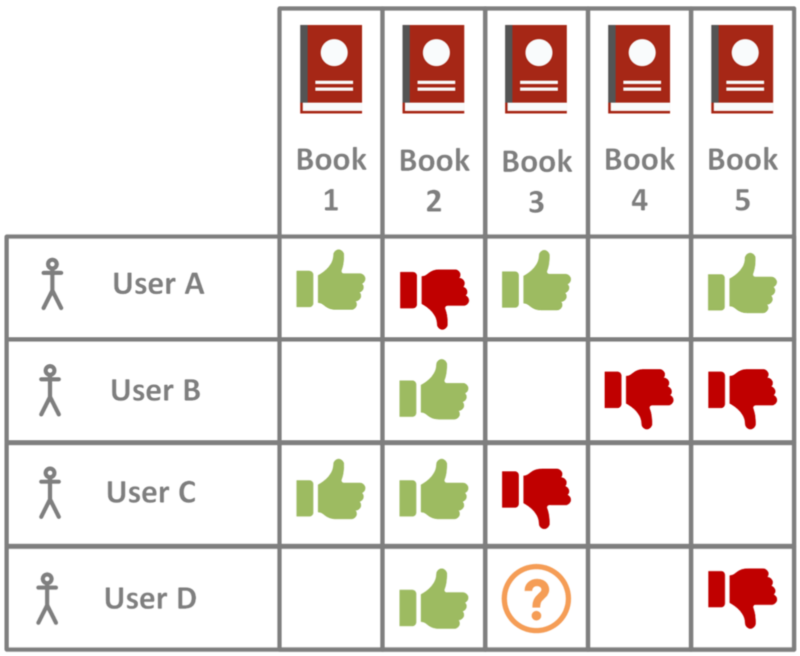
PROJECT 3 REPORT

Collaborative Filtering



Fabrice Harel-Canada (705221880)
Ganesha Srivallabha Durbha (405291327)
Boyuan He (004791432)

2021.02.26

21W-ECENGR219-1

Q1

Compute the sparsity of the movie rating dataset

ANSWER

```
Sparsity is 0.016999683055613623

Num of available ratings =  100836

Num of possible ratings =  5931640
```

The R matrix is of size (610, 9724),ie,

```
Number of users: 610

Number of movies: 9724
```

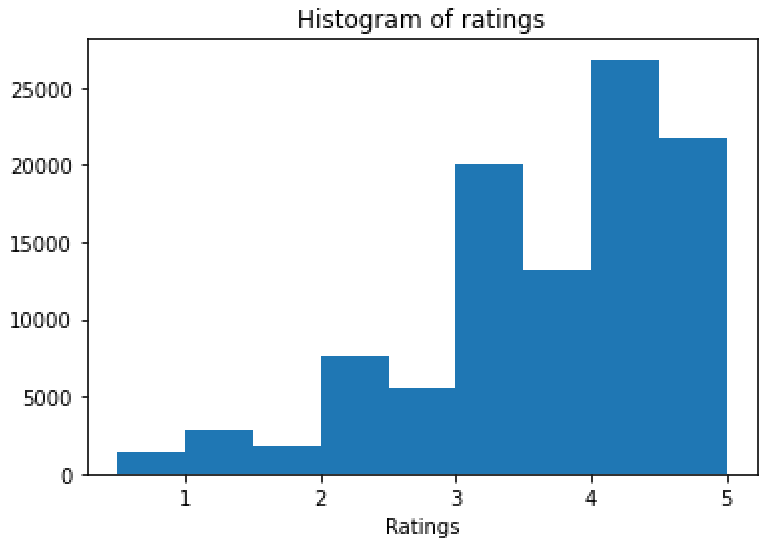
The first five rows of R look like:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userId																					
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Q2

Plot a histogram showing the frequency of the rating values.

ANSWER



Most of the ratings fall between 3 and 5. The distribution of ratings is not centered about the center of the ratings domain [0.5,5].

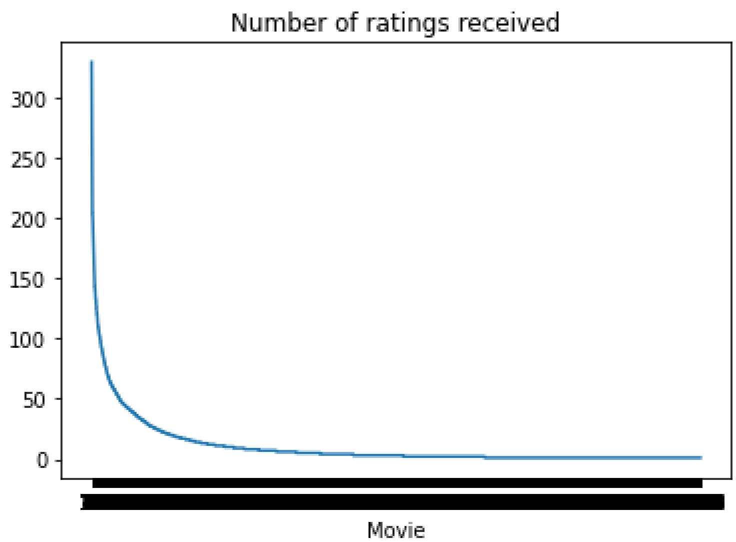
That means users tended to give high ratings and this factor must be considered while developing a learning algorithm.

Q3

Distribution of the number of ratings received among movies

ANSWER

A monotonically decreasing curve is expected:

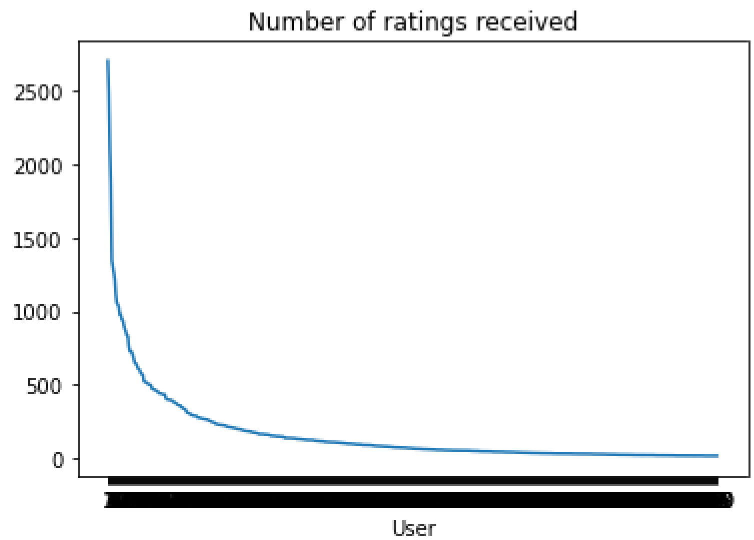


Q4

Distribution of ratings among users

ANSWER

A monotonically decreasing curve is expected.



Q5

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

ANSWER

Number of ratings strictly less than 3:
19073 of all the ratings available are less than 3

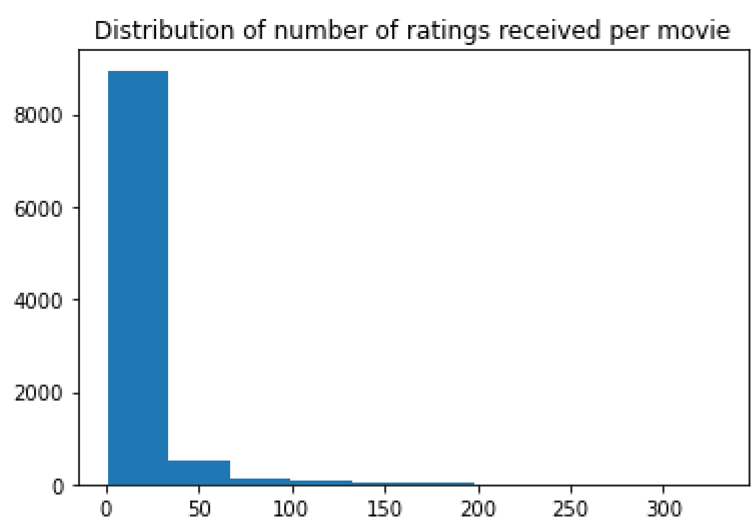
Number of ratings greater than or equal to 3:
81763 of all the ratings available are greater than or equal to 3

Percentage of ratings greater than or equal to 3:
81.085 % of the ratings are greater than or equal to 3

As we see from the statistics and the histogram of Q3, the ratings distribution is skewed in that most of the ratings received are greater than or equal to three. By calculation, 81.085% of the ratings received are greater than or equal to 3. Thus, movies were given high ratings in general.

To know a user's preference for a movie, it would be more informative to find the difference of a user's rating from their mean ratings. Positive values in this case tell us that the user liked the movie more than what they feel to be an average movie.

Let us also look at the number of ratings received per movie.



Number of movies that received less than 25 ratings:

8712

Median number of ratings received by a movie

3.0

Mean of number of ratings per movie:

10.369806663924312

Movie with the maximum number of ratings:
329

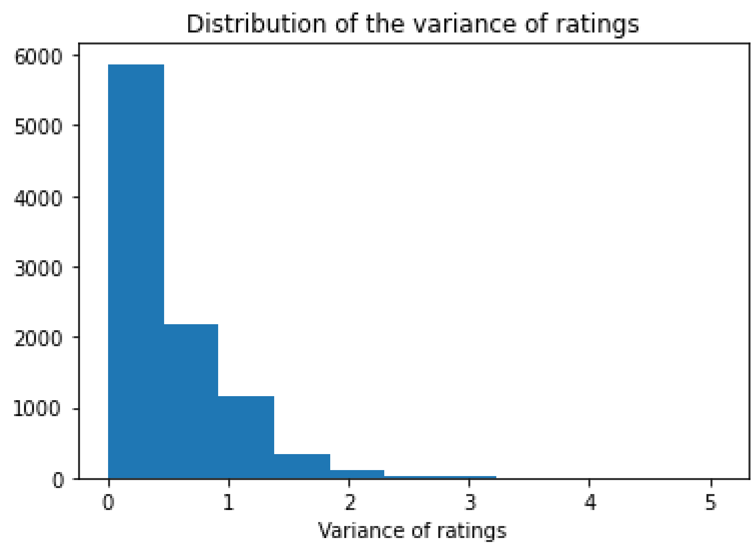
A histogram of the number of ratings received per movie shows that many movies received less than 25 ratings (8712/9724 = 89.59% of the movies received less than 25 ratings). As less information is available about many movies, the testing has to be performed with this consideration. The median number of ratings received per movie is 3, Showing that at least 50% of the movies have at most three ratings.

Q6

Histogram of the variance of the rating values received by each movie

ANSWER

Maximum variance is 5.0625
Minimum variance is 0.0



Notice the high number of movies with low variance in their ratings. Almost all movies (more than 7000) had their ratings variance less than 1. Apart from high ratings received by good movies across user tastes, one other reason is the small scale in ratings(0.5 to 5 in steps of 0.5), which limits the gap between the maximum and minimum possible rating value.

It must be noted that despite these reasons, if a movie still has large variance in the ratings it received, it would be better to test the model on such movies separately.

Q7

Pearson Correlation Coefficient

ANSWER

Following the given notation, the equation for the Pearson Correlation Coefficient is

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{size(I_u)}$$

Q8

ANSWER

$$I_u \cap I_v$$

Indicates the set of movie indices of the common movies rated by both user u and user v.

$$I_u \cap I_v = \phi$$

Indicates that there were no common movies rated by both user u and user v.

Q9

Can you explain the reason behind mean-centering the raw ratings ($r_{vj} - \mu_v$) in the prediction function?

ANSWER

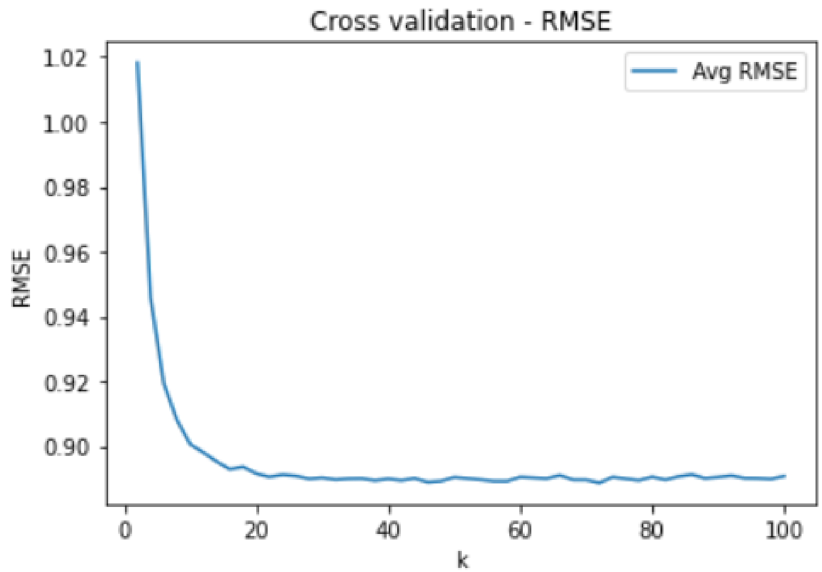
Since ratings ought to help us compare how much a user liked one movie over the others they watched, mean centered ratings are a better measure for this purpose. This is because mean centering removes the individual bias of users - some users give raw ratings in the higher ranges for all movies, and hence mean centering allows us to conclude that the mean centered positive rated movies are the ones they prefer more than an average movie according to them.

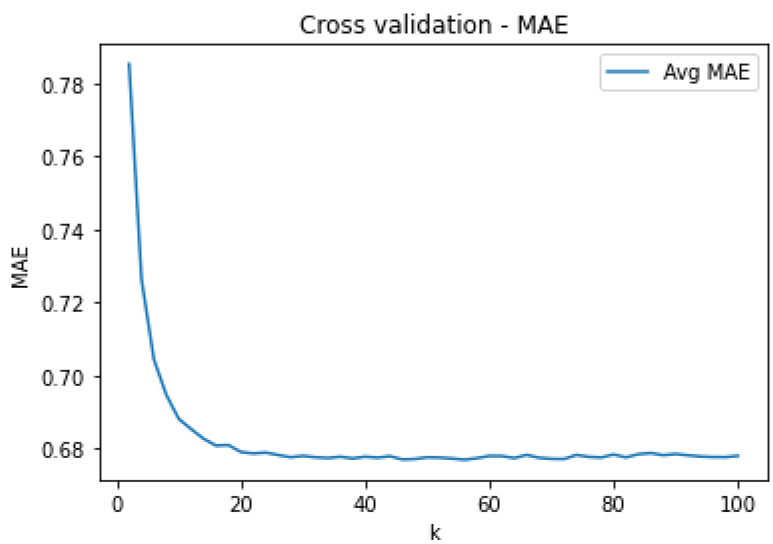
Q 10

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

ANSWER

	Avg RMSE	Avg MAE
2	1.018185	0.785315
4	0.945780	0.726422
6	0.919274	0.704402
8	0.907985	0.694670
10	0.900566	0.688128
12	0.898031	0.685315
14	0.895245	0.682683
16	0.892838	0.680807
18	0.893598	0.680957
20	0.891548	0.679053
22	0.890492	0.678661
24	0.891178	0.678899
28	0.889896	0.677627
30	0.890224	0.677997
32	0.889658	0.677616
34	0.889924	0.677401
36	0.889983	0.677771
38	0.889405	0.677288
40	0.889895	0.677780
42	0.889430	0.677497
44	0.890063	0.677914
46	0.888887	0.676999
48	0.889206	0.677133
50	0.890377	0.677605
52	0.890017	0.677512
52	0.890017	0.677512
54	0.889651	0.677271
56	0.889203	0.676932
58	0.889188	0.677363
60	0.890454	0.677981
62	0.890219	0.677948
64	0.889975	0.677395
66	0.890996	0.678260
68	0.889610	0.677476
70	0.889597	0.677185
72	0.888651	0.677122
74	0.890439	0.678246
76	0.889922	0.677737


Cross validation - RMSE
Avg RMSE
RMSE
1.02
1.00
0.98
0.96
0.94
0.92
0.90
0
20
40
60
80
100
k


Cross validation - MAE
Avg MAE
MAE
0.78
0.76
0.74
0.72
0.70
0.68
0
20
40
60
80
100
k

Q11

Use the plot from question 10, to find a ’minimum k’. Note: The term ’minimum k’ in this context means that increasing k above the minimum value would not result in a significant decrease in average RMSE or average MAE.

ANSWER:

For RMSE, minimum k = 20, and RMSE(k=20) = 0.891

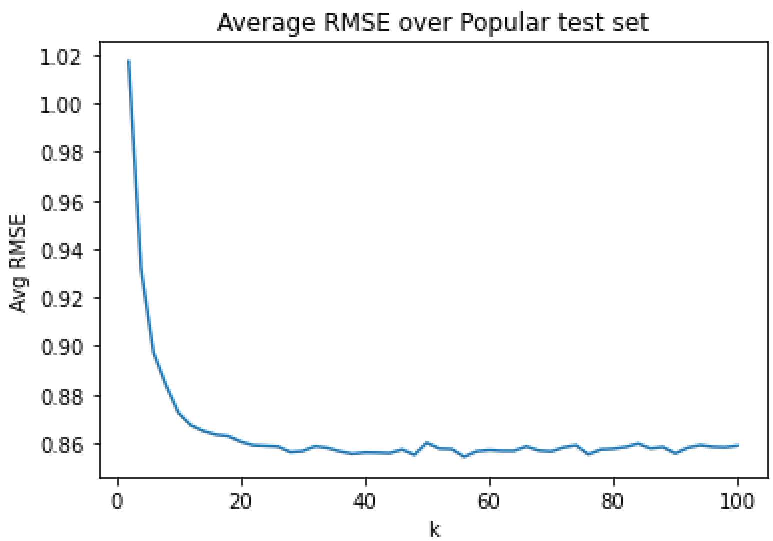
For MAE, minimum k = 28, and MAE (k=28) = 0.677 (note: k = 20 (MAE = 0.679) also gives the stable value approximated until the second decimal)

Q 12

Design a k-NN collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it's performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.

ANSWER:

Minimum average RMSE is 0.8542364591375305

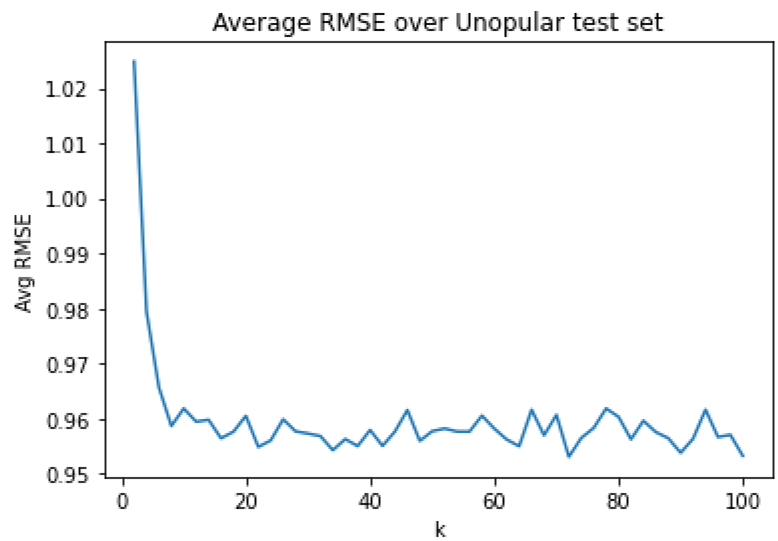


Q 13

Design a k-NN collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate it's performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.

ANSWER:

Minimum average RMSE is 0.9530988852354133

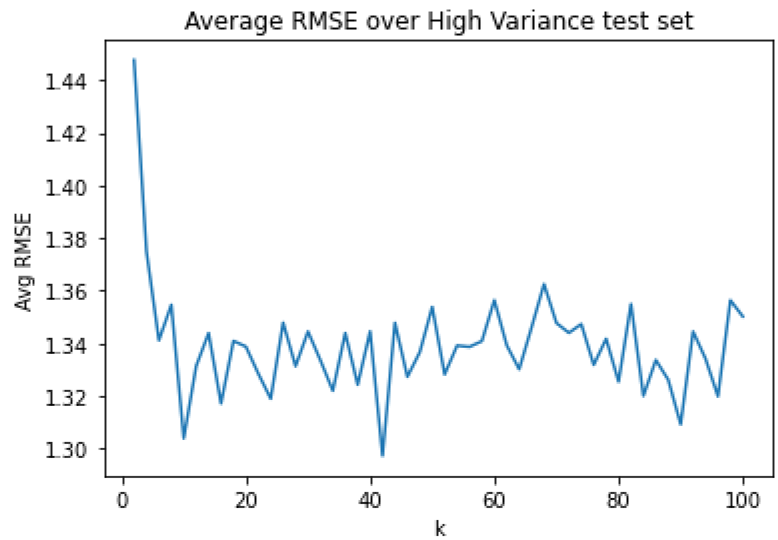


Q 14

Design a k-NN collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate it's performance using 10-fold cross validation. Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds.

ANSWER:

Minimum average RMSE is 1.297356189327368

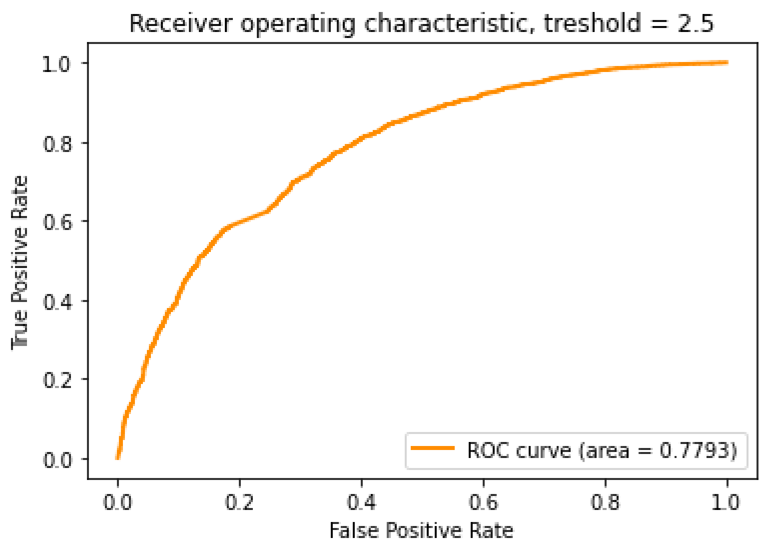


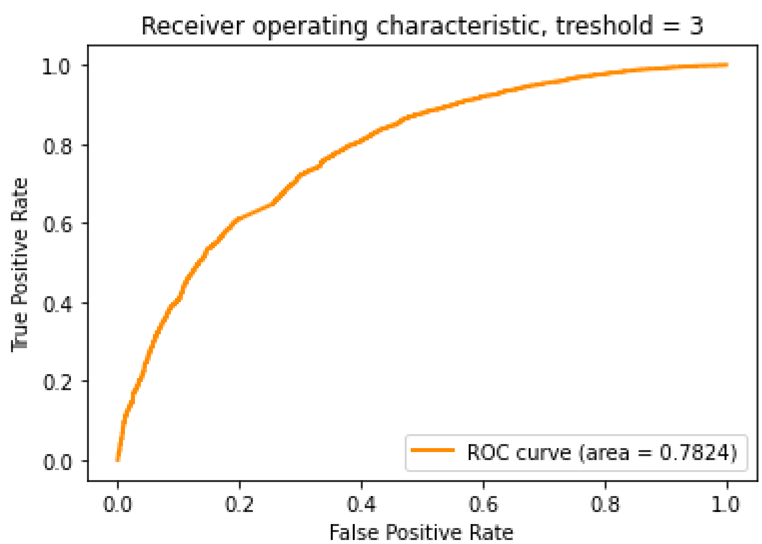
Q 15

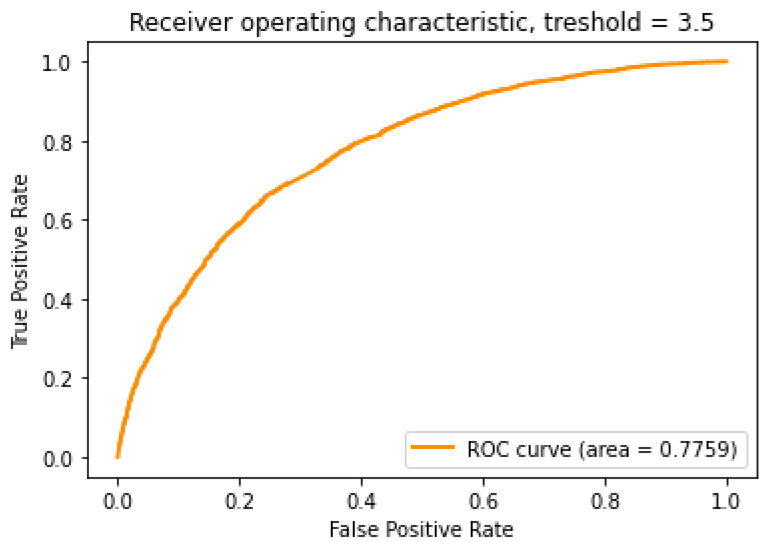
Plot the ROC curves for the k-NN collaborative filter designed in question 10 for threshold values [2.5, 3, 3.5, 4]

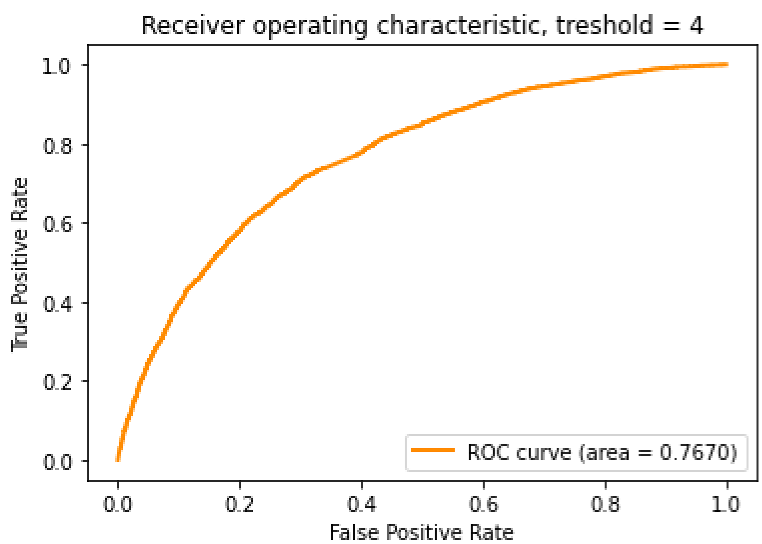
ANSWER:

The area under the curve for each graph is mentioned at bottom right of the image.






Receiver operating characteristic, treshold = 3.5
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (area = 0.7759)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate


Receiver operating characteristic, treshold = 4
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (area = 0.7670)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

Q16

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.

ANSWER

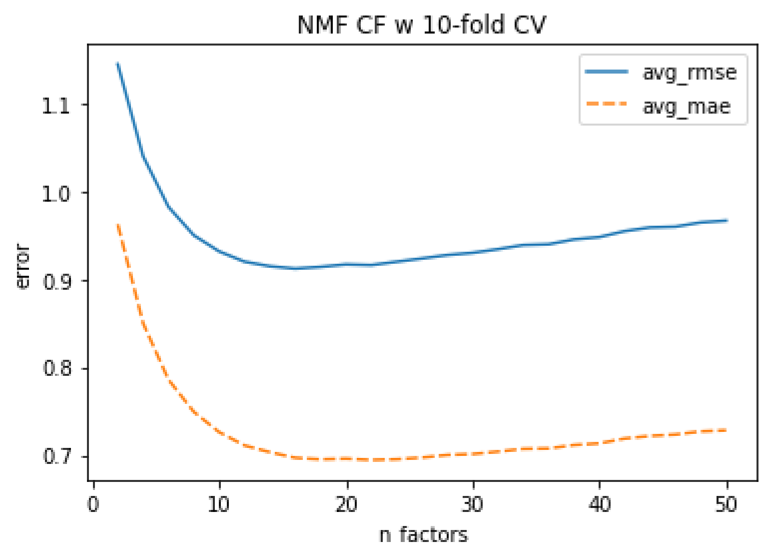
No, it is not convex. For a fixed U, the problem simply minimizes V instead:

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

Q17

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

ANSWER



Q18

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

ANSWER

```
Minimum Average RMSE: 0.912127 @ k=16
Minimum Average MAE:  0.694464 @ k=22
```

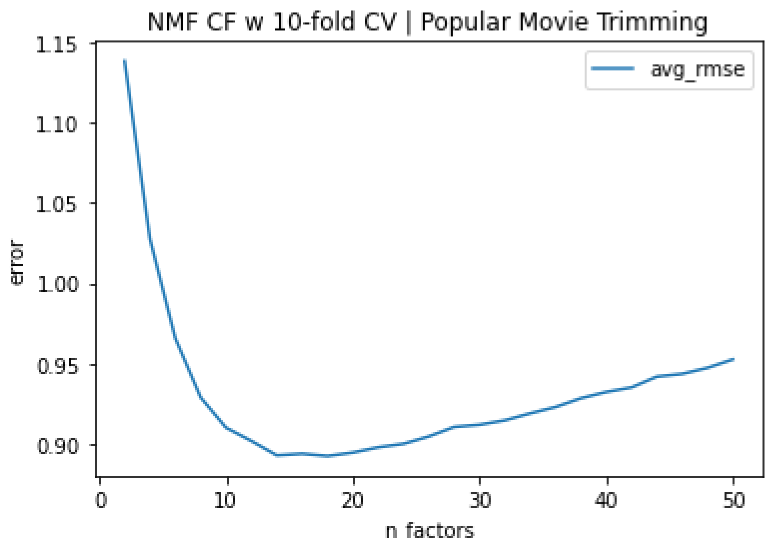
The $k_{min} \in \{16, \quad 22\}$ do seem to roughly correspond to the 18 tracked movie genres.

NOTE: I excluded `(no genres listed)` and `IMAX` from the total genre count because these do not seem to be valid. The former is the `NULL` class and it is likely that a genre could be assigned by some expert. The latter is just a type of theater / movie format.

Q19

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

ANSWER

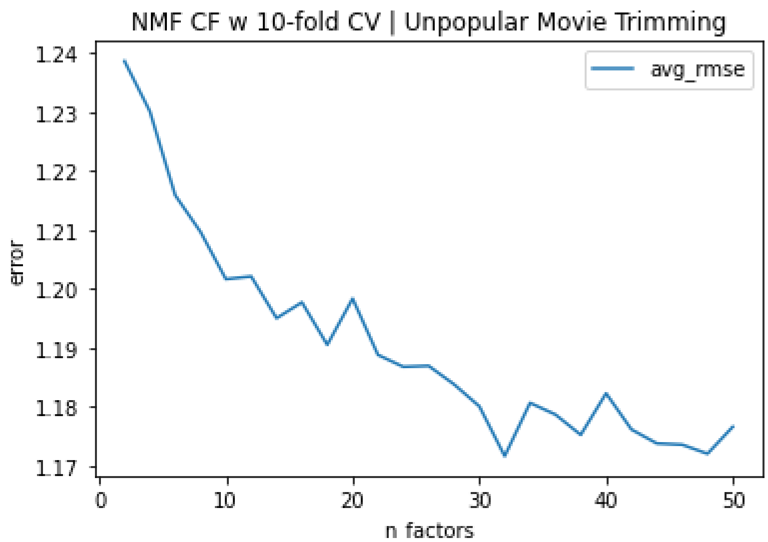



```
Minimum Average RMSE: 0.892921 @ k=18
```

Q20

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

ANSWER

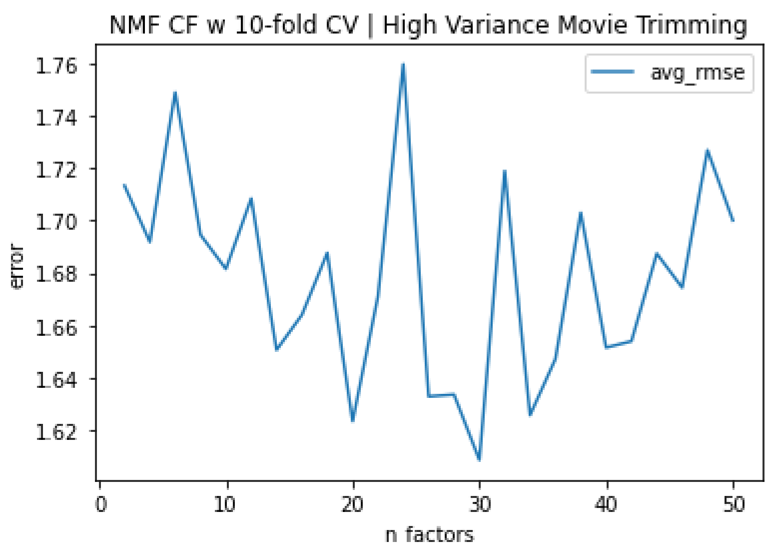



```
Minimum Average RMSE: 1.171689 @ k=32
```

Q21

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

ANSWER

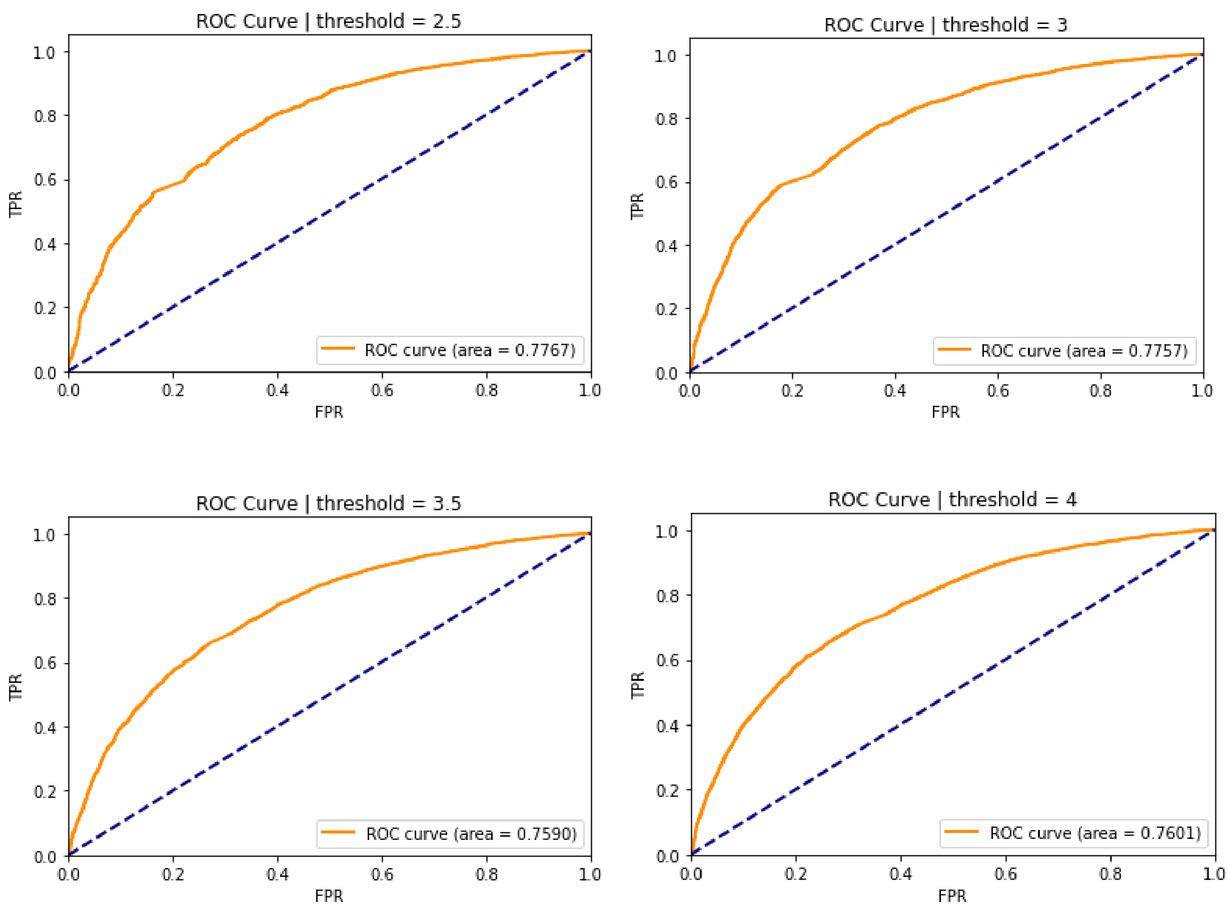



```
Minimum Average RMSE: 1.60866 @ k=30
```

Q22

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

ANSWER



Q23

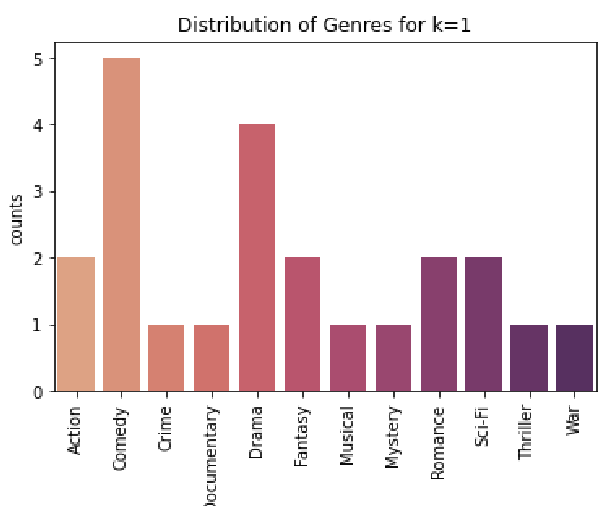
Perform Non-negative matrix factorization on the ratings matrix R to obtain the factor matrices U and V, where U represents the user-latent factors interaction and V represents the movie-latent factors interaction (use k = 20). For each column of V, sort the movies in descending order and report the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

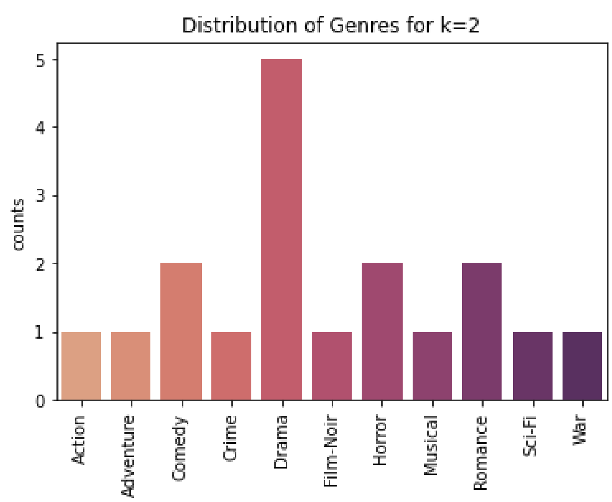
ANSWER

Each latent factor (LF) seems to correspond to a relatively small collection of genres, but they aren't particularly distinctive. For each LF, we plotted the distribution of movie genre assignments to see which were the most prevalent. The lowest number of unique genres was 7 and the highest was 14. The most number of movies belonging to the same genre for a given LF was 7 out of 10, and 4 out of 10 for the lowest.

As for the connection between the LFs and the genres, there does seem to be a weak connection between them. For example LF={19} shows a strong presence of thriller, LF={1,2,8,12} for comedy, but many others show the dominance of drama in the genre assignments.

Below is a set of genre distributions for various dimensions of V. Please see the jupyter notebook for all such distributions. NOTE: k in this context is a dimension of V.



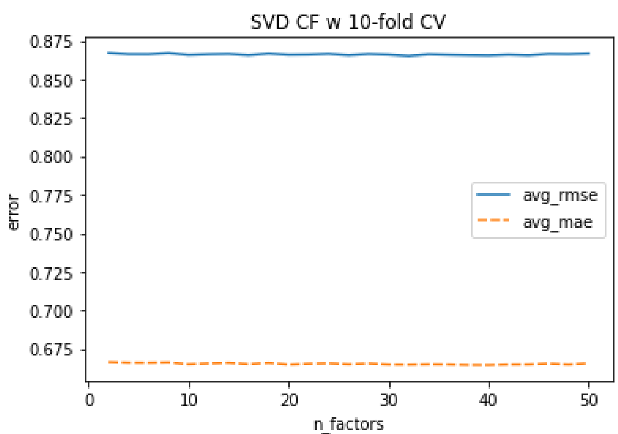


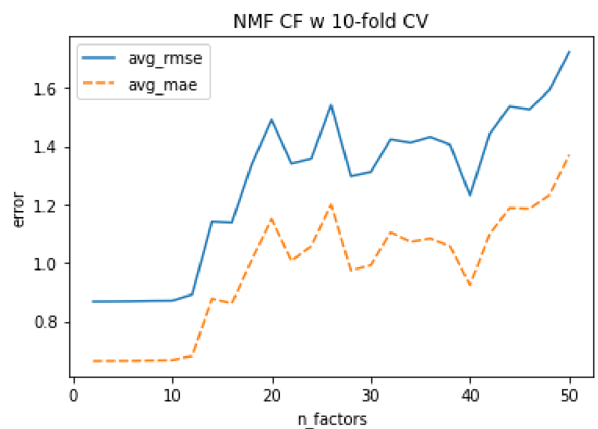
Q24

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

ANSWER

NOTE: The question suggested we use SVD with bias, but we also calculated NMF with bias and will present both results in conjunction. We initially did this just to see if NMF got better results when calculated with bias and the results are mixed. The min RMSE for SVD w Bias is lower for SVD but higher for MAE. NMF w Bias also has a higher AUC (shown later). Either way, the results are pretty close and we hope our inclusion of more work than what was originally requested is well received.





Q25

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

ANSWER

SVD w Bias:

```
Minimum Average RMSE: 0.865094 @ k=32
Minimum Average MAE:  0.664435 @ k=40
```

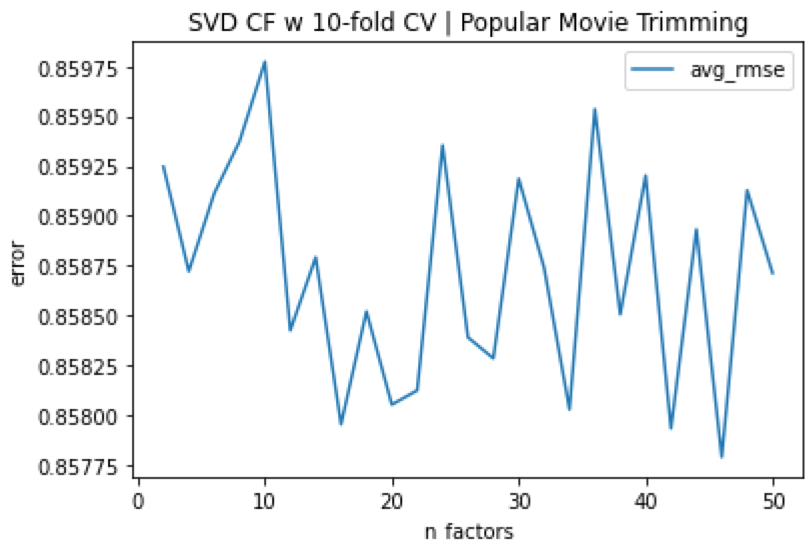
NMF w Bias:

```
Minimum Average RMSE: 0.867691 @ k=2
Minimum Average MAE:  0.664269 @ k=2
```

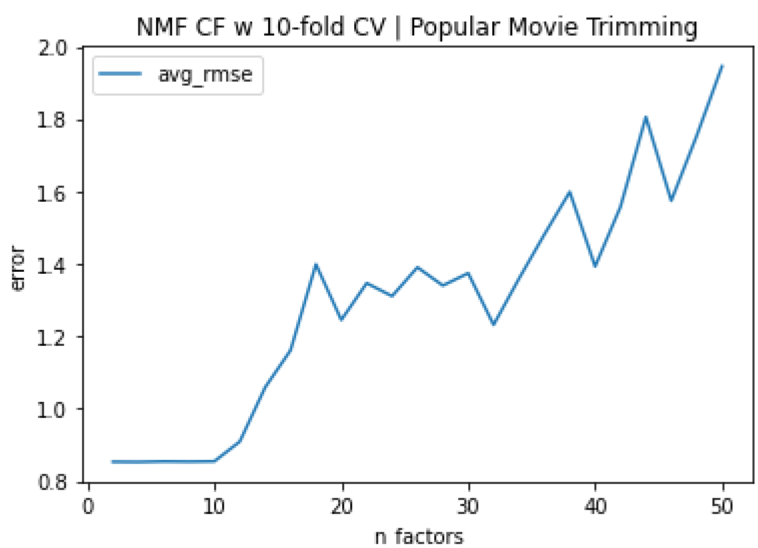
Q26

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

ANSWER




```
Minimum Average MAE:  0.85779 @ k=46
```

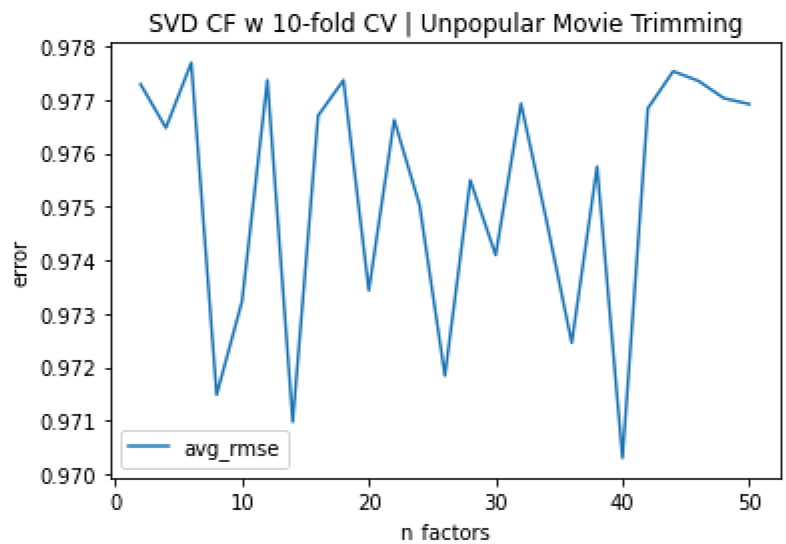



```
Minimum Average MAE:  0.853147 @ k=4
```

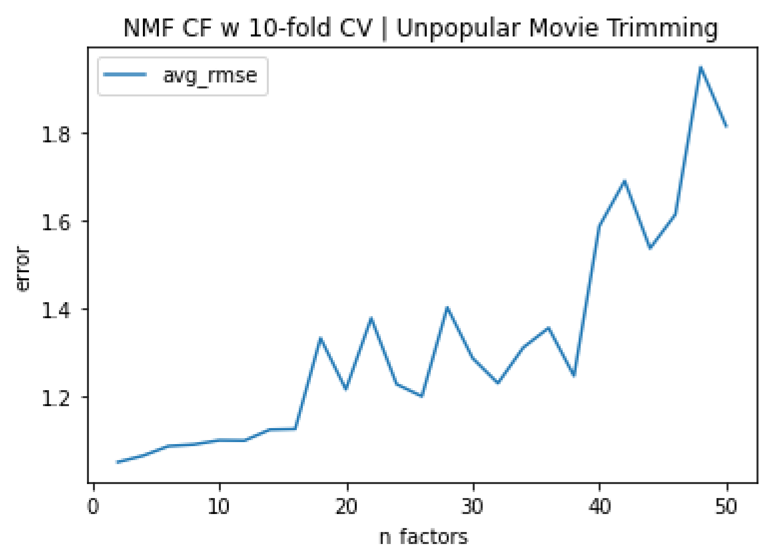
Q27

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

ANSWER




```
Minimum Average MAE:  0.970299 @ k=40
```

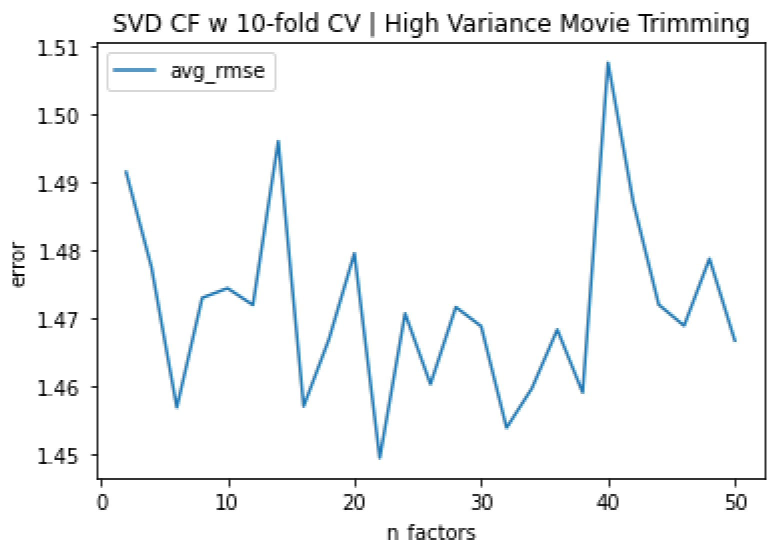



```
Minimum Average MAE:  1.052906 @ k=2
```

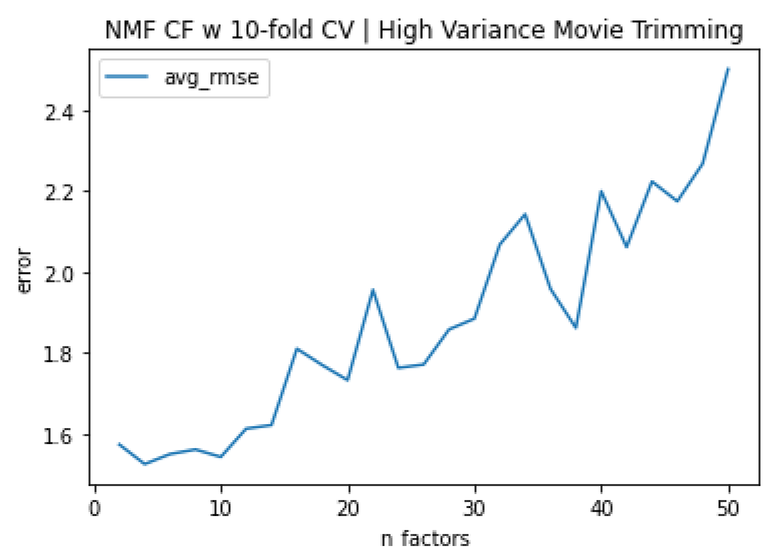
Q28

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

ANSWER




```
Minimum Average MAE:  1.449506 @ k=22
```



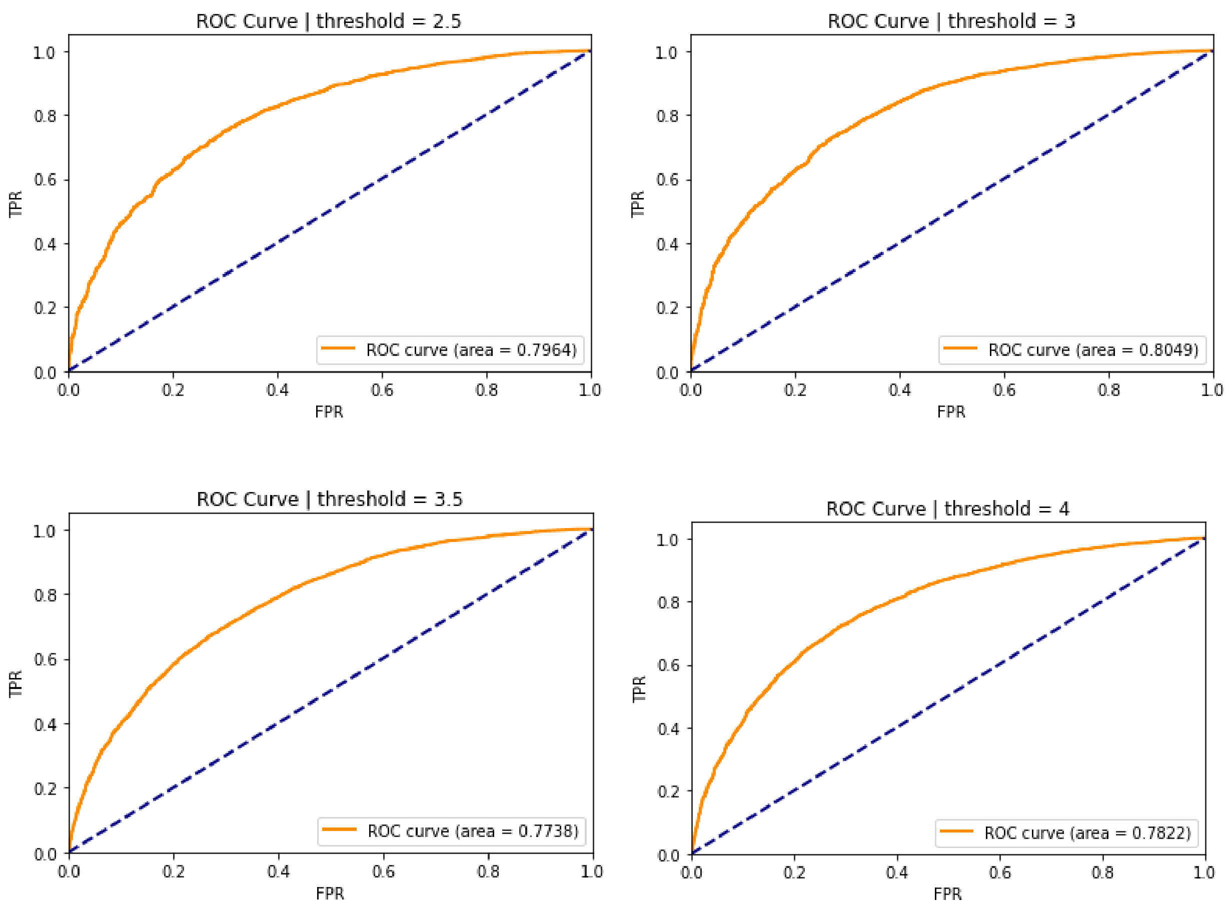

```
Minimum Average MAE:  1.524563 @ k=4
```

Q29

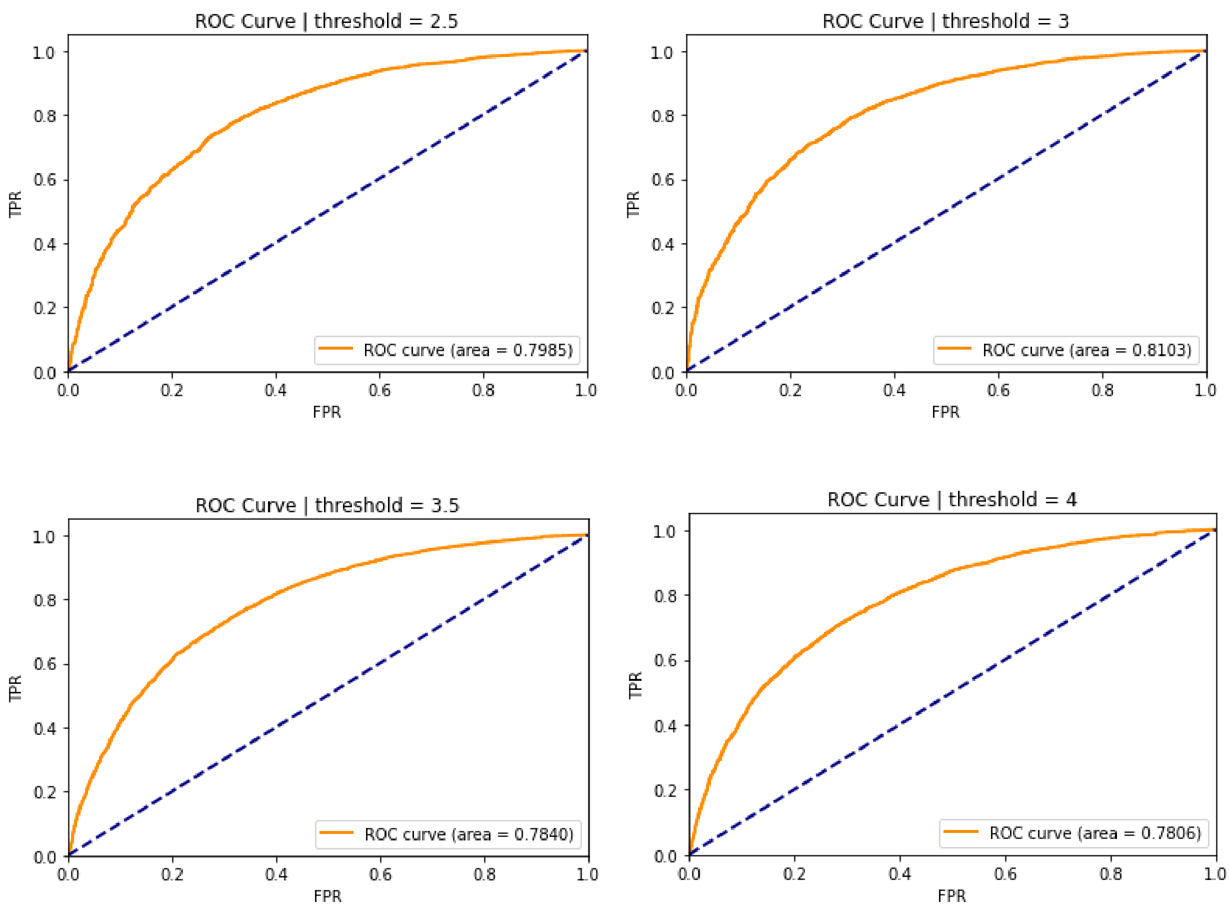
Plot the ROC curves for the MF with bias collaborative filter designed in question 24 for threshold values [2.5, 3, 3.5, 4]. For the ROC plotting use the optimal number of latent factors found in question 25. For each of the plots, also report the area under the curve (AUC) value.

ANSWER

SVD w Bias @ k=32



NMF w Bias @ k=2


ROC Curve | threshold = 2.5
ROC curve (area = 0.7985)
ROC Curve | threshold = 3
ROC curve (area = 0.8103)
ROC Curve | threshold = 3.5
ROC curve (area = 0.7840)
ROC Curve | threshold = 4
ROC curve (area = 0.7806)
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0

Q30

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

ANSWER

As mentioned in the project manual: for Naive collaborative filtering

The predicted rating of user i for item j, denoted by $\hat{r}_{ij}$ is given by equation 11

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

where μ_i is the mean rating of user i.

Here we report the average RMSE of 10 fold cross validation using the test dataset. The RMSE score is 9.347089292911079.

Q31

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

ANSWER

We apply the naive collaborative filter on popular movie trimmed test sets, and the resulting average RMSE score over 10 fold cross validation is 9.323127210045904.

Q32

Design a naive collaborative filter to predict the ratings of the movies in the unpopular movie trimmed test set and evaluate it's performance using 10-fold cross validation.

ANSWER

We apply the naive collaborative filter on unpopular movie trimmed test sets, and the resulting average RMSE score over 10 fold cross validation is 9.710962250099168.

Q33

Design a naive collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate it's performance using 10-fold cross validation

ANSWER

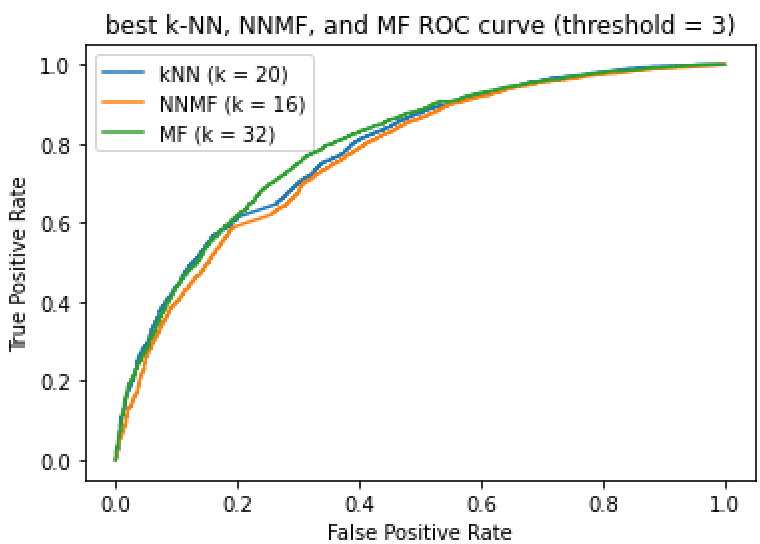
We apply the naive collaborative filter on high variant movie trimmed test sets, and the resulting average RMSE score over 10 fold cross validation is 9.350126702356985.

Q34

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

ANSWER

We show the ROC curve using k-NN (k=20), NNMF(k=16), and MF(k=32) with bias based collaborative filters (threshold set to 3). As we can see in the figure below, MF with bias based collaborative filters slightly outperform k-NN and NNMF. It has the largest area under the ROC curve, which means it produces better movie rating predictions.



Q35

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.

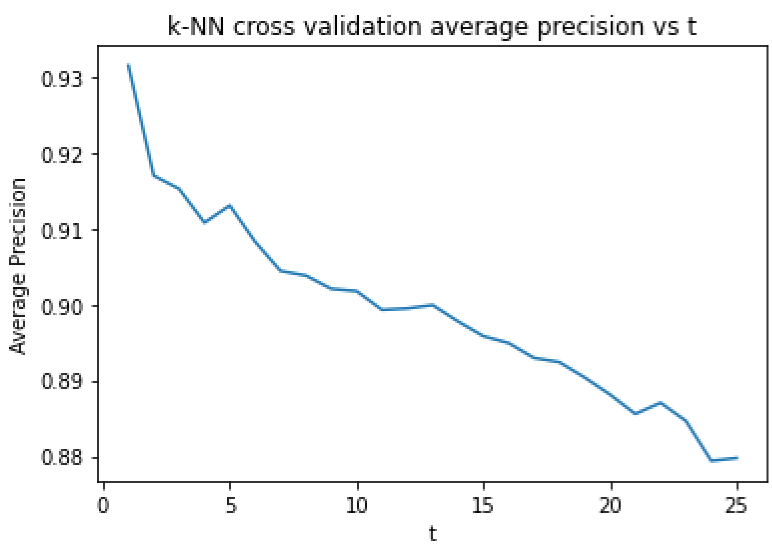
ANSWER

Precision is the fraction of liked and recommended items over the whole recommendation Recall is the fraction of liked and recommended items over everything liked.

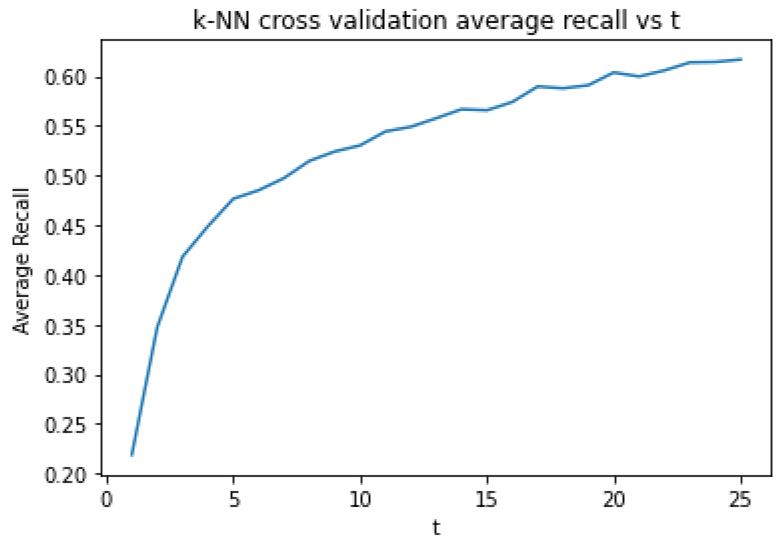
Q36

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

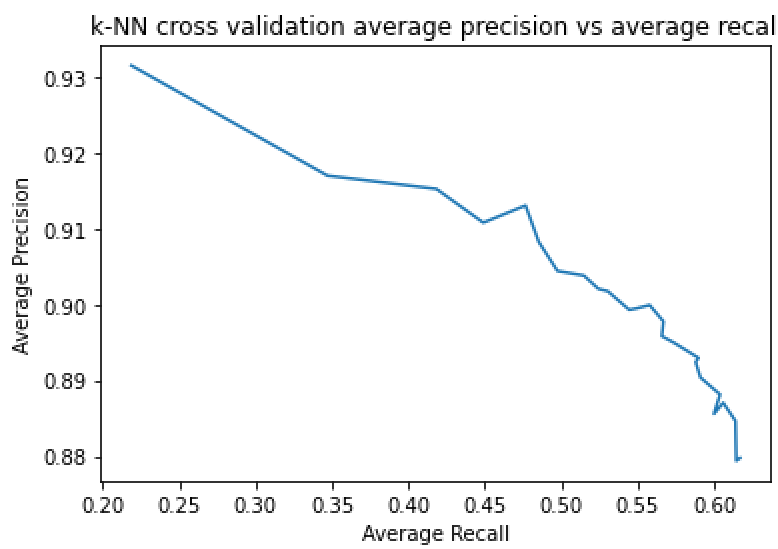
ANSWER



As we can see, precision and t have an negative correlation, which means precision score gets lower as we increase t



Recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger)

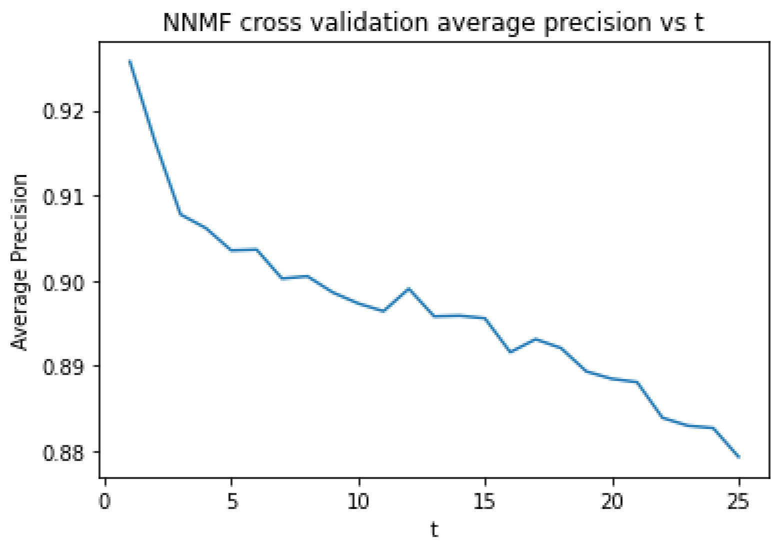


Precision and recall have negative correlation, which means precision score is lower when recall score is higher

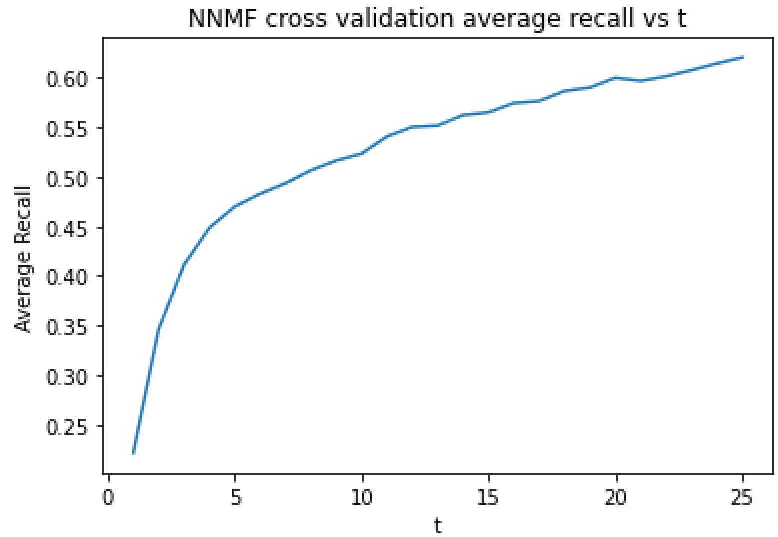
Q37

Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using NNMF-based collaborative filter predictions. Also, plot the average recall (Y-axis) against t (X-axis) and average precision (Y-axis) against average recall (X-axis). Use the optimal number of latent factors found in question 18 and sweep t from 1 to 25 in step sizes of 1. For each plot, briefly comment on the shape of the plot.

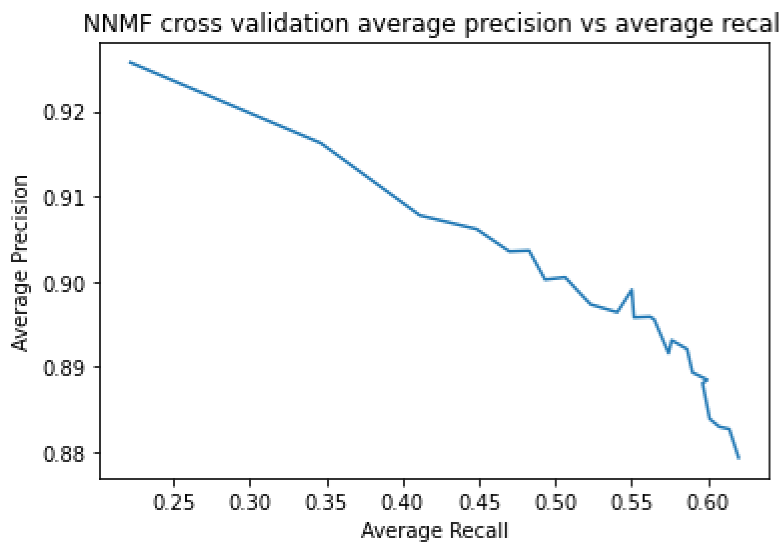
ANSWER



Similar to KNN result, precision and t have an negative correlation, which means precision score gets lower as we increase t



Recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger)

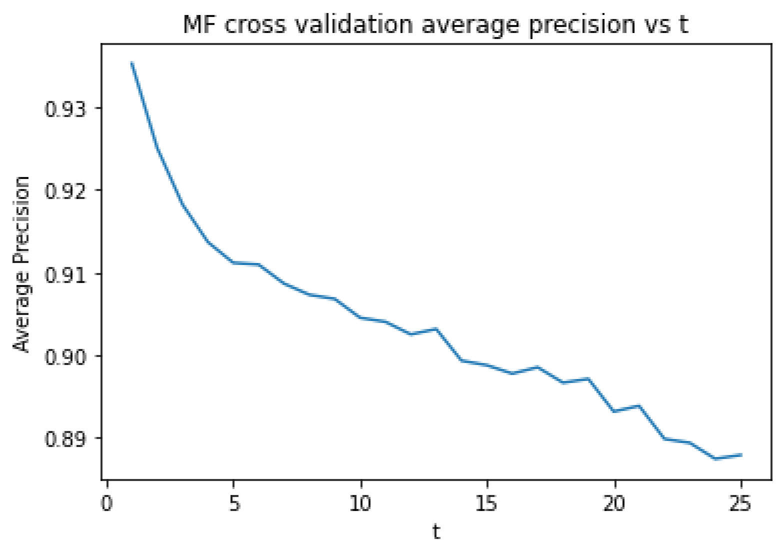


Precision and recall have negative correlation, which means precision score is lower when recall score is higher

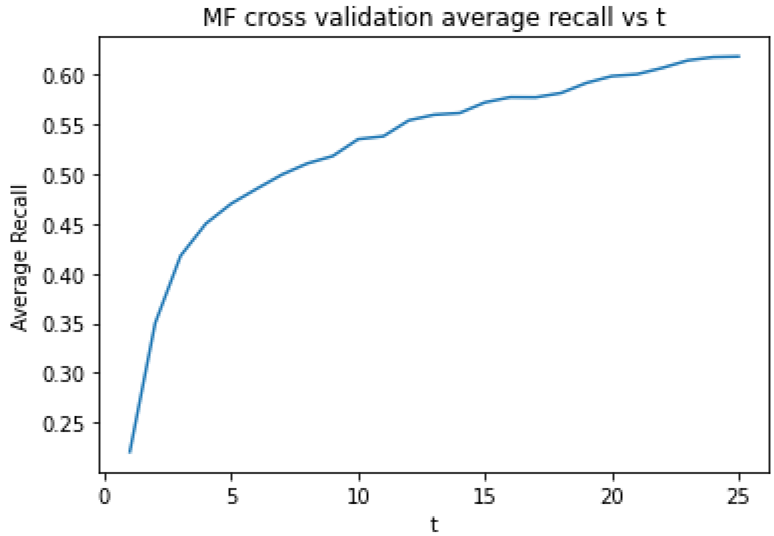
Q38

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

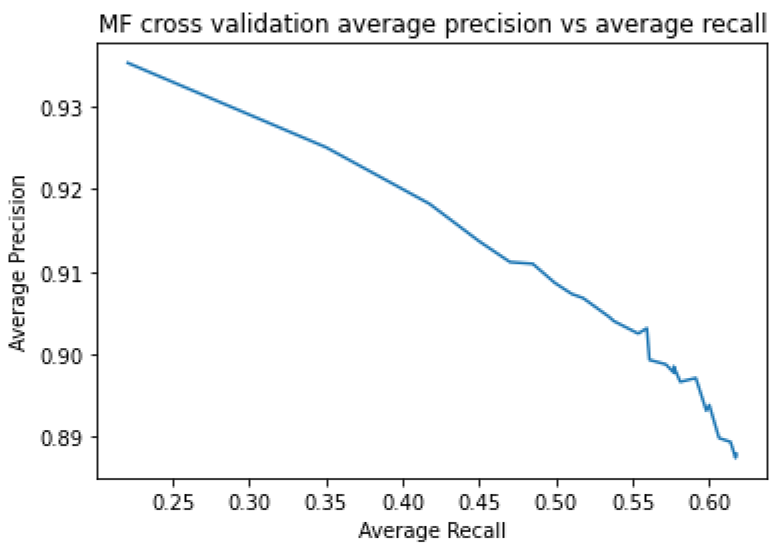
ANSWER



Similar to previous two result, precision and t have an negative correlation, which means precision score gets lower as we increase t



Recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger)

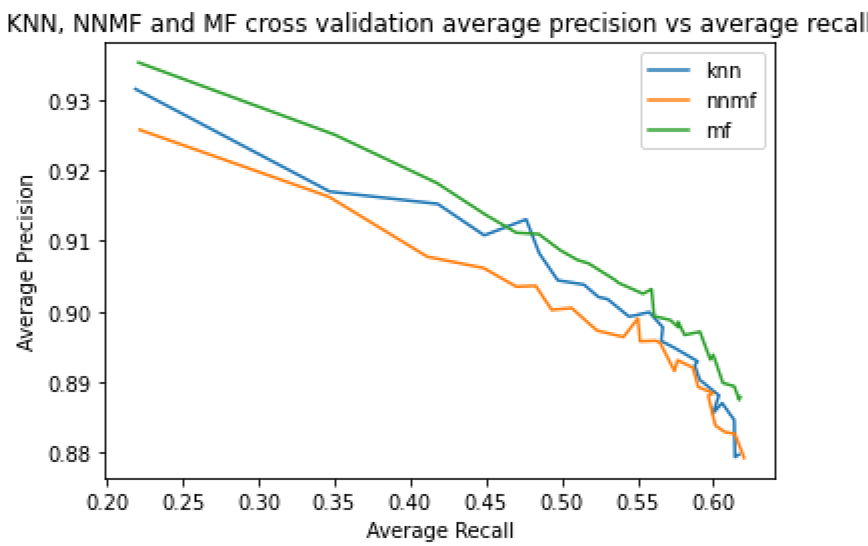


Precision and recall have negative correlation, which means precision score is lower when recall score is higher

Q39

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

ANSWER



Here we show the precision-recall curve of k-NN, NNMF, and MF. As we can see, MF with a bias-based collaborative filter's curve is slightly above the others', which means MF would have better precision and recall score in almost all given t values. Thus we can conclude that MF with a bias-based collaborative filter allows us to produce the most relevant recommendations.

Code

Following pages contain jupyter notebook code for the project, which is roughly divided into 3 parts (question 1-15, 16-29 and 30-39). Each part has detailed explanations and comments.

In [1]:
```
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import random
import string
import sys

import warnings
warnings.filterwarnings('ignore')

np.set_printoptions(precision=4, suppress=True)

RANDOM_SEED = 42

np.random.seed(RANDOM_SEED)
random.seed(RANDOM_SEED)
```

In [2]:
```
cols = ["userId", "movieId", "rating"]
Rdf = pd.read_csv('C:/Work/UCLA/Winter 2021/219 Large Scale Data Mining Models and Algorithms/Project_3/ml
#Rdf = pd.read_csv('C:/Work/UCLA/Winter 2021/219 Large Scale Data Mining Models and Algorithms/Project_3/m
R = Rdf.pivot(index = "userId", columns = "movieId", values = "rating")
R.head()
```

Out[2]:

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	1935
userId																	
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N

5 rows × 9724 columns

In [3]:
```
R = R.to_numpy()  #R is now the numpy Ratings matrix
m = R.shape[0]      #m is the number of users
n = R.shape[1]      #n is the number of movies

print("Number of users:", m)
print("Number of movies:", n)
```

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

Q1. Sparsity

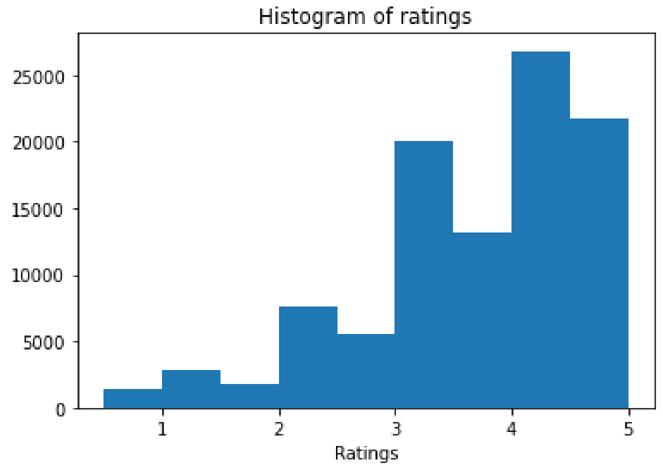
In [4]:
```
num_ratings = Rdf.shape[0]
sparsity = num_ratings/(m*n)

print("Sparsity is", sparsity)
print("Num of available ratings = ", num_ratings)
print("Num of possible ratings = ", m*n)
```

```
Sparsity is 0.016999683055613623
Num of available ratings =  100836
Num of possible ratings =  5931640
```

Q2. Histogram showing frequency of rating values

In [5]:

```
plt.hist(Rdf["rating"], bins = [0.5,1,1.5,2,2.5,3,3.5,4,4.5,5])
plt.title("Histogram of ratings")
plt.xlabel("Ratings")
plt.show()
```

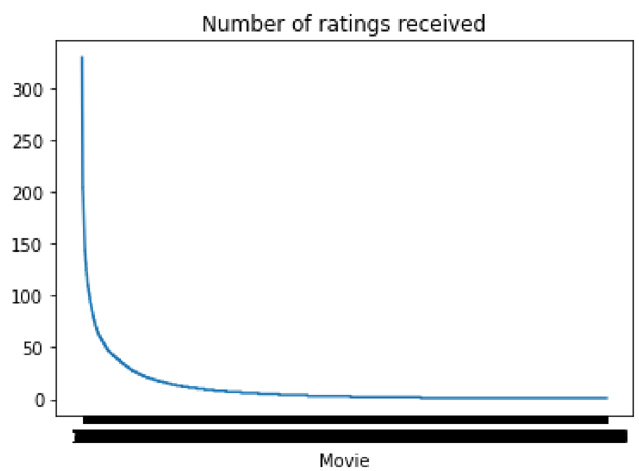


Most of the ratings fall between 3 to 5. The distribution of ratings is not centered about the center of the ratings domain [0.5,5]. That means users have tended to give high ratings and this factor must be considered while developing a learning algorithm.

Q3. Distribution of number of ratings received among movies

In [83]:

```
mov = Rdf['movieId'].value_counts()
mov = mov.to_numpy()

plt.plot(np.arange(1,n+1), mov)
plt.xticks(np.arange(1,n+1))
plt.title("Number of ratings received")
plt.xlabel("Movie")
plt.show()
```

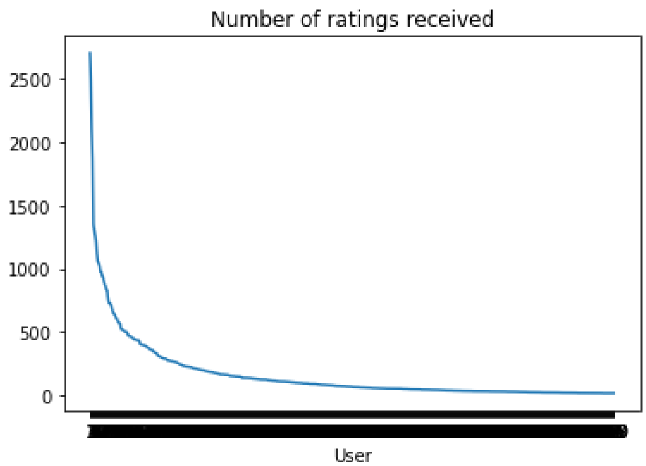


Q4. Distribution of the number of ratings among users

In [84]:

```
users = Rdf['userId'].value_counts()
users = users.to_numpy()

plt.plot(np.arange(1,m+1), users)
```

```
plt.xticks(np.arange(1,m+1))
plt.title("Number of ratings received")
plt.xlabel("User")
plt.show()
```

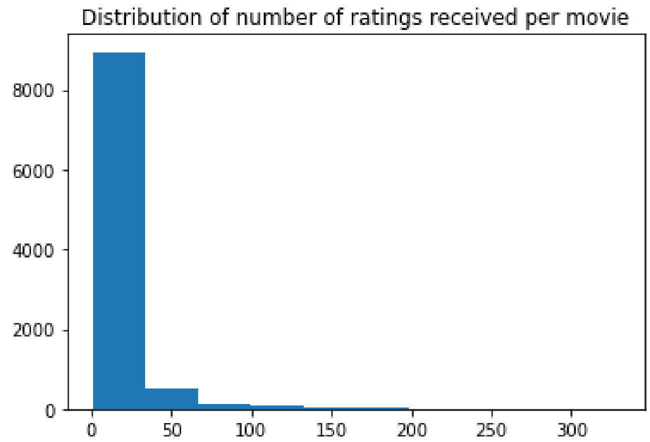


Q5. Features of the ratings distribution

In [28]:

```
print("Number of ratings strictly less than 3:")
nrl = len(Rdf[Rdf['rating'] < 3])
print(nrl)

print("Number of ratings greater than or equal to 3:")
nrh = len(Rdf[Rdf['rating'] >= 3])
print(nrh)

print("Percentage of ratings greater than or equal to 3:")
print(100.0 * nrh/(nrh+nrl))

permov_rat = Rdf.value_counts(["movieId"])
print("Number of movies with ratings less than or equal to 25:")
print(len(permov_rat[permov_rat <= 25]))

M_mov = permov_rat.to_numpy()
plt.hist(M_mov)
plt.title("Distribution of number of ratings received per movie")
plt.show()

print("Median number of ratings received by a movie")
print(np.median(M_mov))
print("Mean of number of ratings per movie:")
print(np.mean(M_mov))
print("Movie with the maximum number of ratings:")
print(np.max(M_mov))
```

```
Number of ratings strictly less than 3:
19073
Number of ratings greater than or equal to 3:
81763
Percentage of ratings greater than or equal to 3:
81.08512832718473
Number of movies with ratings less than or equal to 25:
8712
```




```
Median number of ratings received by a movie
3.0
Mean of number of ratings per movie:
10.369806663924312
Movie with the maximum number of ratings:
329
```

As we see from the statistics and the histograms, the ratings distribution is skewed in that most of the ratings are greater than or equal to three. By calculation, 81.085% of the ratings recevied are greater than or equal to 3. Thus, most users tended to give high ratings and only few users tended to give numerically low ratings.

To know a user's preference for a movie, it would be more informative to find the difference of a user's rating form their mean ratings. Positive values in this case tell us that the user liked the move more than an average movie according to them.

Also, a histogram of the number of ratings received per movie shows that many movies received less than 25 ratings (8712/9724 = 89.59% of the movies received less than 25 ratings). As less information is available about many movies, the testing has to be performed with this consideration. The medain number of ratings received per per movie is 3, Showing that atleast 50% of the movies have atmost three ratings.

Some features of the distributions are:

Number of ratings strictly less than 3: 19073 Number of ratings greater than or equal to 3: 81763 Percentage of ratings greater than or equal to 3: 81.08512832718473

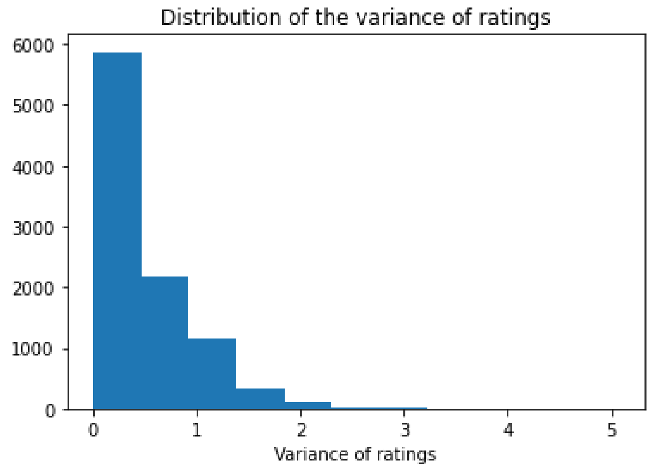
Number of movies with ratings less than or equal to 25: 8712 Median number of ratings received by a movie 3.0 Mean of number of ratings per movie: 10.369806663924312

Q6. Histogram of variance of the rating values received by each movie

In [85]:

```
mov_var = Rdf[["movieId","rating"]].groupby(['movieId']).var(ddof=0) #Ref: ddof = 0 makes the divisor usec
mov_var = mov_var.to_numpy()
mov_var = np.squeeze(mov_var)

print("Maximum variance is", np.max(mov_var))
print("Minimum variance is",np.min(mov_var))
bins = np.ceil((np.max(mov_var) - np.min(mov_var))/0.5)
#print(bins)
plt.hist(mov_var, bins = int(bins))
plt.title("Distribution of the variance of ratings")
plt.xlabel("Variance of ratings")
plt.show()
```

```
Maximum variance is 5.0625
Minimum variance is 0.0
```



Notice the high number of movies with low variance in their ratings. Almost all movies (greater than 7000) had their ratings variance less than 1. Apart from high ratings received by good movies across user tastes, one other reason for is the small scale in ratings(0.5 to 5 in steps of 0.5). Thus movies that still have large variance must be tested separately.

Q7. Pearson Correlation Coefficient

$$\mu_u = \frac{\Sigma_{k \in I_u} r_{uk}}{size(I_u)}$$

Q8.

$I_u \cap I_v = \phi$ indicates that there were no common movies rated by both user u and user v

Q9. Mean centering

Since ratings ought to help us compare how much a user liked one movie over the others they watched, mean centered ratings are a better measure for this purpose. This is because mean centering removes the individual bias of users - some users give raw ratings in the higher ranges for all movies, and hence mean centering allows us to conclude that the now positive rated movies are the ones they prefer more than an average movie according to them.

In [86]:
```
#conda install -c conda-forge scikit-surprise
```

Q 10.

In [6]:
```
from surprise import Reader
from surprise import Dataset
from surprise.prediction_algorithms.knns import KNNWithMeans
from surprise import similarities
from surprise.model_selection import cross_validate
from surprise.model_selection import KFold
from surprise import accuracy
```

In [88]:
```
reader = Reader(rating_scale=(0.5, 5))
R_data = Dataset.load_from_df(Rdf[["userId", "movieId", "rating"]], reader)

sim_options = {'name': 'pearson',
               'user_based': True}

k_list = np.arange(2,101,2)
scores = []

for k in k_list:
```

```
    algo = KNNWithMeans(k = k,sim_options=sim_options, verbose = False)
    pred = cross_validate(algo, R_data, cv = 10, verbose = False, n_jobs = -1)
    rmse = np.mean(pred['test_rmse'])
    rmae = np.mean(pred['test_mae'])
    scores.append([rmse, rmae])

scores_df = pd.DataFrame(scores, columns = ['Avg RMSE', 'Avg MAE'], index = k_list)

scores_df
```

Out[88]:

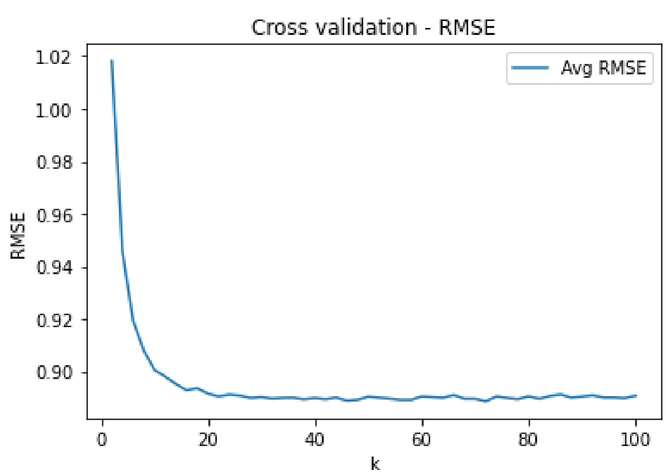
	Avg RMSE	Avg MAE
2	1.018185	0.785315
4	0.945780	0.726422
6	0.919274	0.704402
8	0.907985	0.694670
10	0.900566	0.688128
12	0.898031	0.685315
14	0.895245	0.682683
16	0.892838	0.680807
18	0.893598	0.680957
20	0.891548	0.679053
22	0.890492	0.678661
24	0.891178	0.678899
26	0.890790	0.678243
28	0.889896	0.677627
30	0.890224	0.677997
32	0.889658	0.677616
34	0.889924	0.677401
36	0.889983	0.677771
38	0.889405	0.677288
40	0.889895	0.677780
42	0.889430	0.677497
44	0.890063	0.677914
46	0.888887	0.676999
48	0.889206	0.677133
50	0.890377	0.677605
52	0.890017	0.677512
54	0.889651	0.677271
56	0.889203	0.676932
58	0.889188	0.677363
60	0.890454	0.677981
62	0.890219	0.677948
64	0.889975	0.677395
66	0.890996	0.678260

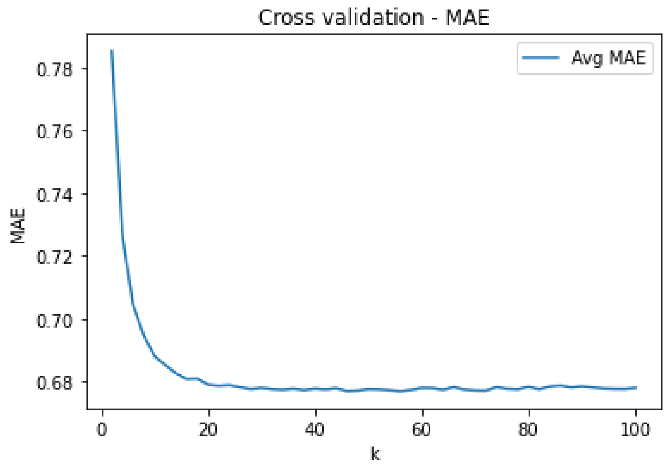
	Avg RMSE	Avg MAE
68	0.889610	0.677476
70	0.889597	0.677185
72	0.888651	0.677122
74	0.890439	0.678246
76	0.889922	0.677737
78	0.889441	0.677554
80	0.890499	0.678370
82	0.889595	0.677582
84	0.890622	0.678465
86	0.891233	0.678771
88	0.890041	0.678171
90	0.890417	0.678509
92	0.890909	0.678145
94	0.890091	0.677830
96	0.890037	0.677685
98	0.889833	0.677653
100	0.890720	0.678004

In [89]:

```
scores_df.plot(y = 'Avg RMSE', xlabel = "k", ylabel = "RMSE", title = "Cross validation - RMSE")
scores_df.plot(y = 'Avg MAE', xlabel = "k", ylabel = "MAE", title = "Cross validation - MAE")
```

Out[89]: <AxesSubplot:title={'center':'Cross validation - MAE'}, xlabel='k', ylabel='MAE'>





Q 11.

For RMSE, minimum k = 20, and RMSE(k=20) = 0.891

For MAE, minimum k = 28, and MAE (k=28) = 0.677 (note: k = 20 (MAE = 0.679) also gives the stable value approximayted until the second decimal)

Q 12.

In [7]:
```
reader = Reader(rating_scale=(0.5, 5))

R_data = Dataset.load_from_df(Rdf[["userId", "movieId", "rating"]], reader)

sim_options = {'name': 'pearson',
               'user_based': True}
```

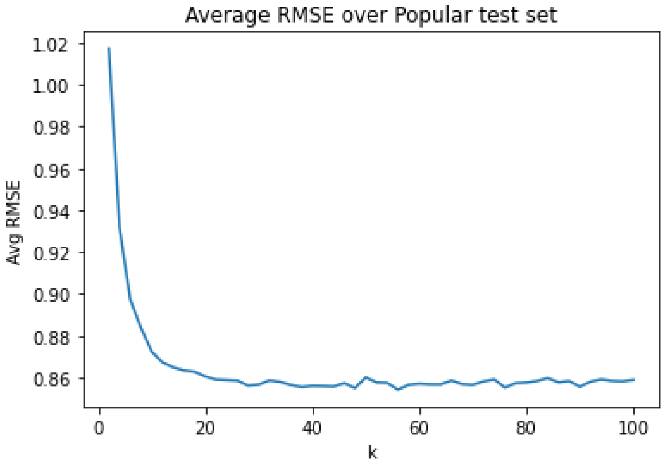
In [5]:
```
def popular_movies(df):
    ts = df.value_counts('movieId') #ts is a series showing number of ratings for each 'movieId'
    movies = ts[ts>2].index #ts[ts>2]'s indices are movieId values satisfying the condition
    trim = df.loc[df['movieId'].isin(movies)]
    trim
    return trim
```

In [6]:
```
kf = KFold(n_splits=10)

k_list = np.arange(2,101,2)

scores = []

for k in k_list:

    algo = KNNWithMeans(k = k,sim_options=sim_options, verbose = False)
    set_scores = []
    for trainset, testset in kf.split(R_data):
        #trainset and testset are lists made of tuples
        algo.fit(trainset)

        test_df = pd.DataFrame(testset, columns = ["userId", "movieId", "rating"])#converting testset intc
        test_df = popular_movies(test_df)
        test_tuples = [tuple(x) for x in test_df.to_numpy()] #converting the trimmed test_df into tuples

        predictions = algo.test(test_tuples)

        set_scores.append(accuracy.rmse(predictions, verbose=False))

    scores.append(sum(set_scores)/len(set_scores))
```

In [7]:

```
print("Minimum average RMSE is ", min(scores))
plt.plot(k_list, scores)
plt.xlabel('k')
plt.ylabel("Avg RMSE")
plt.title("Average RMSE over Popular test set")
plt.show()
```

```
Minimum average RMSE is  0.8542364591375305
```



Q 13. Unpopular movies test sets

In [8]:

```
def unpopular_movies(df):
    ts = df.value_counts('movieId') #ts is a series showing number of ratings for each 'movieId'
    movies = ts[ts<=2].index #ts[ts>2]'s indices are movieId values satisfying the condition
    trim = df.loc[df['movieId'].isin(movies)]
    trim
    return trim
```

In [9]:

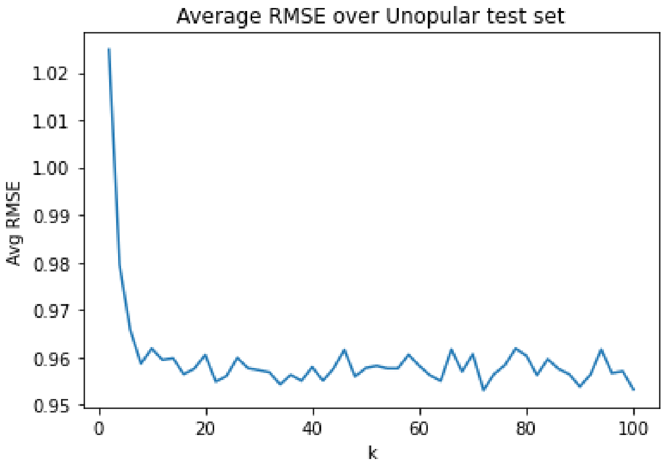
```
kf = KFold(n_splits=10)

k_list = np.arange(2,101,2)

scores = []

for k in k_list:

    algo = KNNWithMeans(k = k,sim_options=sim_options, verbose = False)
    set_scores = []
    for trainset, testset in kf.split(R_data):
        #trainset and testset are lists made of tuples
        algo.fit(trainset)

        test_df = pd.DataFrame(testset, columns = ["userId", "movieId", "rating"])
        test_df = unpopular_movies(test_df)
        test_tuples = [tuple(x) for x in test_df.to_numpy()]

        predictions = algo.test(test_tuples)

        set_scores.append(accuracy.rmse(predictions, verbose=False))

    scores.append(sum(set_scores)/len(set_scores))
```

In [10]:

```
print("Minimum average RMSE is ", min(scores))
plt.plot(k_list, scores)
plt.xlabel('k')
plt.ylabel("Avg RMSE")
```

```
plt.title("Average RMSE over Unopular test set")
plt.show()
```

```
Minimum average RMSE is  0.9530988852354133
```



Q 14. High variance movie trest sets

In [11]:

```
def high_variance(df):

    #trimmed to contain movies with atleast five ratings.
    ts = df.value_counts('movieId')
    movies = ts[ts>=5].index
    df = df.loc[df['movieId'].isin(movies)]

    #consider movies with variance >=2
    mov_var = df[["movieId","rating"]].groupby(['movieId']).var().rename(columns ={'rating':'variance'})
    movies = mov_var[mov_var['variance'] >= 2].index
    df = df.loc[df['movieId'].isin(movies)]
    return df
```

In [12]:

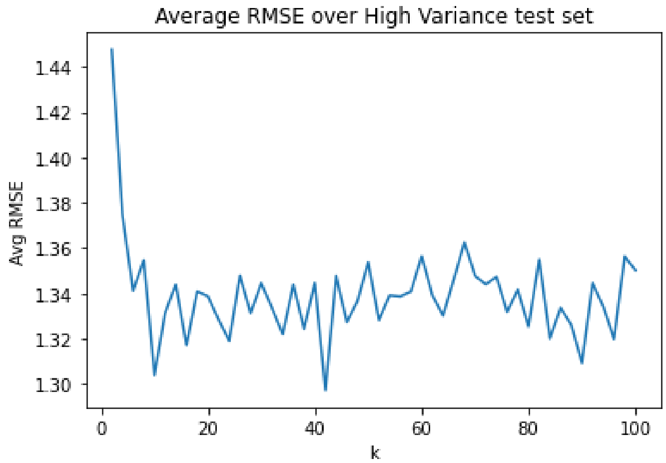
```
kf = KFold(n_splits=10)

k_list = np.arange(2,101,2)

scores = []

for k in k_list:

    algo = KNNWithMeans(k = k,sim_options=sim_options, verbose = False)
    set_scores = []
    for trainset, testset in kf.split(R_data):
        #trainset and testset are lists made of tuples
        algo.fit(trainset)

        test_df = pd.DataFrame(testset, columns = ["userId", "movieId", "rating"])
        test_df = high_variance(test_df)
        test_tuples = [tuple(x) for x in test_df.to_numpy()]

        predictions = algo.test(test_tuples)

        set_scores.append(accuracy.rmse(predictions, verbose=False))

    scores.append(sum(set_scores)/len(set_scores))
```

In [13]:

```
print("Minimum average RMSE is ", min(scores))
plt.plot(k_list, scores)
plt.xlabel('k')
plt.ylabel("Avg RMSE")
```

```
plt.title("Average RMSE over High Variance test set")
plt.show()
```

```
Minimum average RMSE is  1.297356189327368
```



In [ ]:

```

```

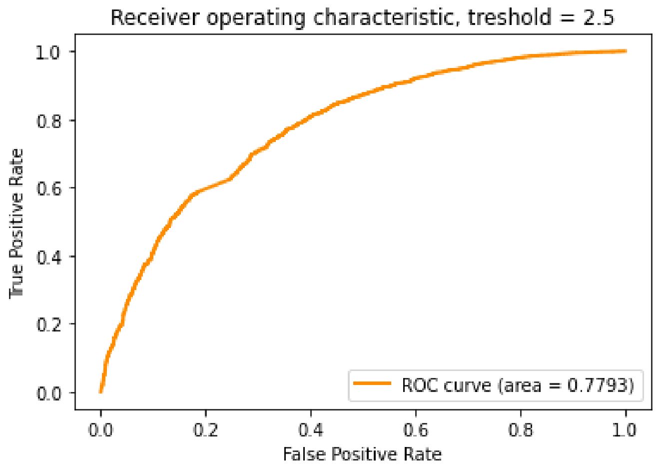
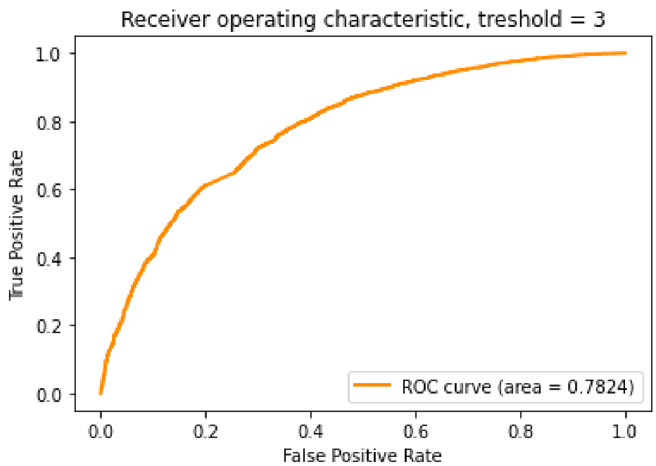
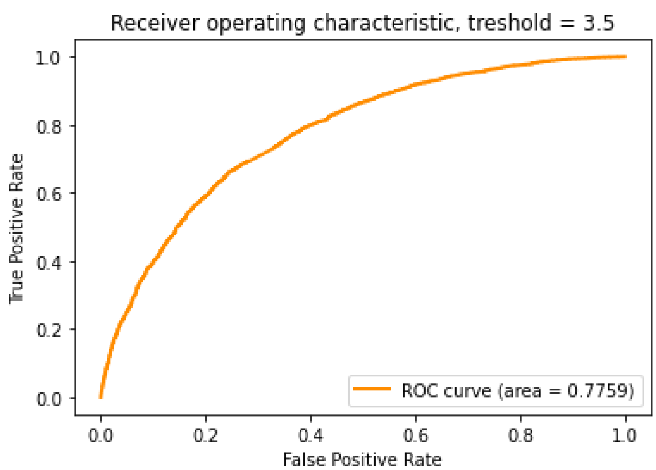
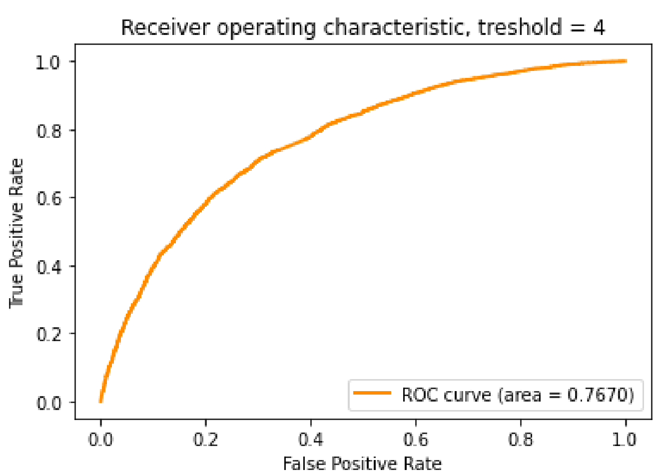
Q 15.

In [8]:

```
from surprise.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
trainset, testset = train_test_split(R_data, test_size=.10)
```

In [11]:

```
thres_list = [2.5, 3, 3.5, 4]

roc_auc = []

for t in thres_list:

    algo = KNNWithMeans(k = 20,sim_options = sim_options, verbose = False)
    algo.fit(trainset)
    preds = algo.test(testset)
    preds = np.asarray(preds)

    true_scores = preds[:,2] >= t #actual scores
    true_scores.astype(int)

    fpr, tpr, tt = roc_curve(true_scores, preds[:,3])
    roc_auc.append(auc(fpr, tpr))

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange',
             lw=2, label='ROC curve (area = %0.4f)' % roc_auc[-1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic, treshold = {}'.format(t))
    plt.legend(loc="lower right")
    plt.show()
```


Receiver operating characteristic, treshold = 2.5
True Positive Rate
False Positive Rate
ROC curve (area = 0.7793)
0.0
0.2
0.4
0.6
0.8
1.0

Receiver operating characteristic, treshold = 3
True Positive Rate
False Positive Rate
ROC curve (area = 0.7824)
0.0
0.2
0.4
0.6
0.8
1.0

Receiver operating characteristic, treshold = 3.5
True Positive Rate
False Positive Rate
ROC curve (area = 0.7759)
0.0
0.2
0.4
0.6
0.8
1.0

Receiver operating characteristic, treshold = 4
True Positive Rate
False Positive Rate
ROC curve (area = 0.7670)
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]:

Project 3: Collaborative Filtering [Q16-29]

ECE 219: Large-Scale Data Mining: Models and Algorithms [Winter 2021]

Prof. Vwani Roychowdhury

UCLA, Department of ECE

Due: 2021.02.19 11:59PM PT

Q16

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.

ANSWER

No, it is not convex. For a fixed U, the problem simply minimizes V instead:

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

Q17

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

In [10]:

```
from surprise.prediction_algorithms.matrix_factorization import NMF
from surprise.model_selection.validation import cross_validate
from surprise.model_selection import train_test_split
from surprise import Dataset, Reader
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

In [13]:

```
reader = Reader(line_format='user item rating timestamp', sep=',',skip_lines=1, rating_scale=(0.5, 5))
file_path = 'ratings.csv'
data = Dataset.load_from_file(file_path, reader=reader)
```

In [17]:

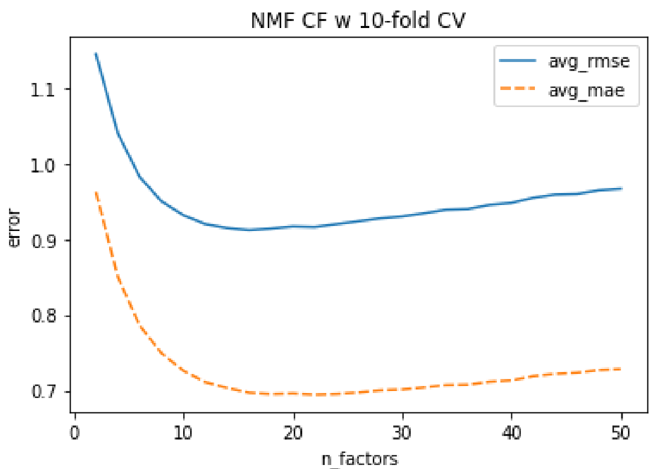
```
ks = np.arange(2, 51, 2)

results = []
for k in ks:
    print('Running with {} factors...'.format(k))
    perf = cross_validate(NMF(n_factors=k), data, cv=10)
    results.append([k, perf['test_rmse'].mean(), perf['test_mae'].mean()])

df = pd.DataFrame(results, columns=['ks', 'avg_rmse', 'avg_mae']).set_index('ks')
```

```
Running with 2 factors...
Running with 4 factors...
```

```
Running with 6 factors...
Running with 8 factors...
Running with 10 factors...
Running with 12 factors...
Running with 14 factors...
Running with 16 factors...
Running with 18 factors...
Running with 20 factors...
Running with 22 factors...
Running with 24 factors...
Running with 26 factors...
Running with 28 factors...
Running with 30 factors...
Running with 32 factors...
Running with 34 factors...
Running with 36 factors...
Running with 38 factors...
Running with 40 factors...
Running with 42 factors...
Running with 44 factors...
Running with 46 factors...
Running with 48 factors...
Running with 50 factors...
```

In [59]:

```
g = sns.lineplot(data=df)
g.set_xlabel('n_factors')
g.set_ylabel('error')
g.set_title('NMF CF w 10-fold CV')
plt.show()
```



Q18

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

ANSWER

```
Minimum Average RMSE: 0.912127 @ k=16
Minimum Average MAE:  0.694464 @ k=22
```

The $k_{min} \in \{16, 22\}$ do seem to roughly correspond to the 18 tracked movie genres.

NOTE: I excluded `(no genres listed)` and `IMAX` from the total genre count because these do not seem to be valid. The former is the `NULL` class and it is likely that a genre could be assigned by some expert. The latter is just a type of theater / movie format.

In [6]:
```
import itertools
```

In [92]:
```
genres = pd.read_csv('movies.csv').genres.tolist()
genres = [x.split('|') for x in genres]
distinct_genres = set(itertools.chain(*genres))
print('Number of Genres: {} \n {}'.format(len(distinct_genres), distinct_genres))
```

```
Number of Genres: 20
 {'Children', 'IMAX', 'Musical', 'Western', 'Horror', 'Action', 'Crime', 'Drama', '(no genres listed)', 'Ro
mance', 'Documentary', 'Fantasy', 'Sci-Fi', 'Comedy', 'Mystery', 'Adventure', 'Film-Noir', 'Thriller', 'Wa
r', 'Animation'}
```

In [72]:
```
df.sort_values('avg_rmse').head(1)
```

Out[72]:

	avg_rmse	avg_mae
ks		
16	0.912127	0.696921

In [73]:
```
df.sort_values('avg_mae').head(1)
```

Out[73]:

	avg_rmse	avg_mae
ks		
22	0.916132	0.694464

Q19

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

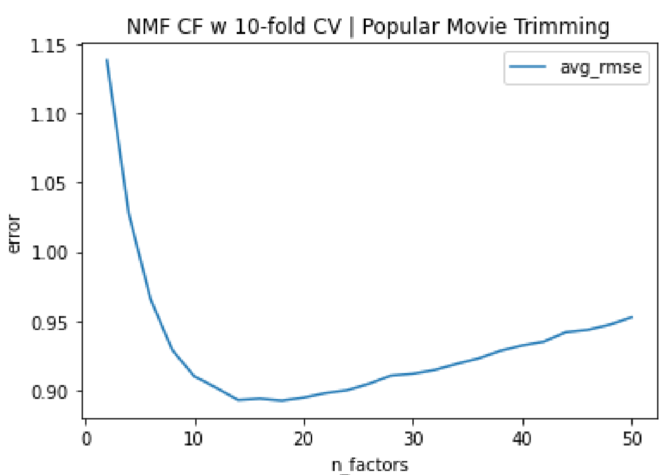
In [7]:
```
from surprise.model_selection import KFold
from surprise import accuracy

from collections import defaultdict
from tqdm.notebook import tqdm
```

In [21]:
```
def MF_plot(MF_type='NMF', trim=None, biased=False, n_folds=10, ks=np.arange(2, 51, 2)):

    reader = Reader(line_format='user item rating timestamp', sep=',',skip_lines=1, rating_scale=(0.5, 5))
    file_path = 'ratings.csv'
    data = Dataset.load_from_file(file_path, reader=reader)

    mvr = defaultdict(list)
    for r in data.raw_ratings:
        mvr[r[1]].append(r[2])

    kf = KFold(n_splits=n_folds)


    results = []
    for k in tqdm(ks):

        if MF_type == 'NMF':
            MF = NMF(n_factors=k, biased=biased)
        elif MF_type == 'SVD':
            MF = SVD(n_factors=k, biased=biased)
```

```
        rmse = 0
        for train, test in kf.split(data):
            if trim == 'Popular':
                test = [r for r in test if len(mvr[r[1]]) > 2]
            elif trim == 'Unpopular':
                test = [r for r in test if len(mvr[r[1]]) <= 2]
            elif trim == 'High Variance':
                test = [r for r in test if (len(mvr[r[1]]) >= 5 and np.var(mvr[r[1]]) >= 2)]
            pred = MF.fit(train).test(test)
            rmse += accuracy.rmse(pred, verbose=False)
        results.append([k, rmse / n_folds])

    df = pd.DataFrame(results, columns=['ks', 'avg_rmse']).set_index('ks')

    g = sns.lineplot(data=df)
    g.set_xlabel('n_factors')
    g.set_ylabel('error')
    g.set_title(MF_type + ' CF w 10-fold CV | ' + trim + ' Movie Trimming')
    plt.show()

    print(df.sort_values('avg_rmse').head(1))
```

In [42…

```
MF_plot(MF_type='NMF', trim="Popular")
```



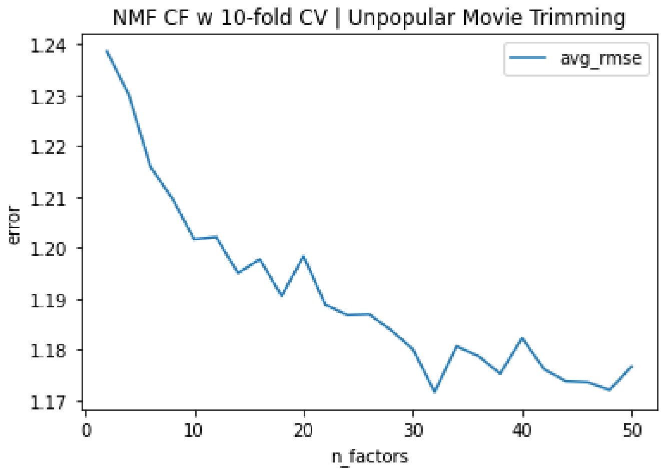

```
    avg_rmse
ks
18  0.892921
```

Q20

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

In [42…

```
MF_plot(MF_type='NMF', trim="Unpopular")
```



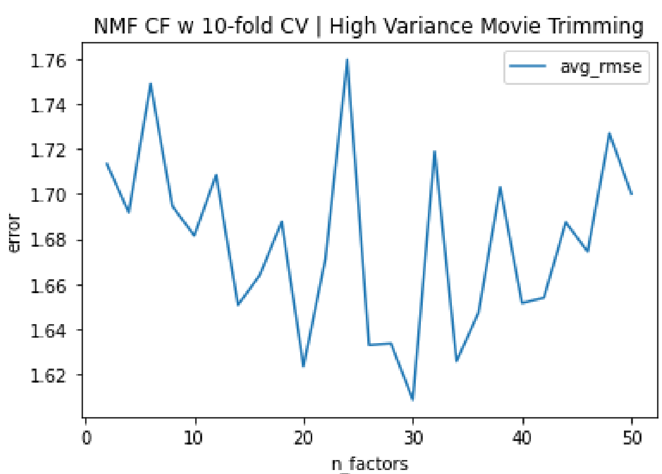

```
    avg_rmse
ks
32  1.171689
```

Q21

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

In [43…

```
MF_plot(MF_type='NMF', trim="High Variance")
```




```
    avg_rmse
ks
30   1.60866
```

Q22

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

In [28]:

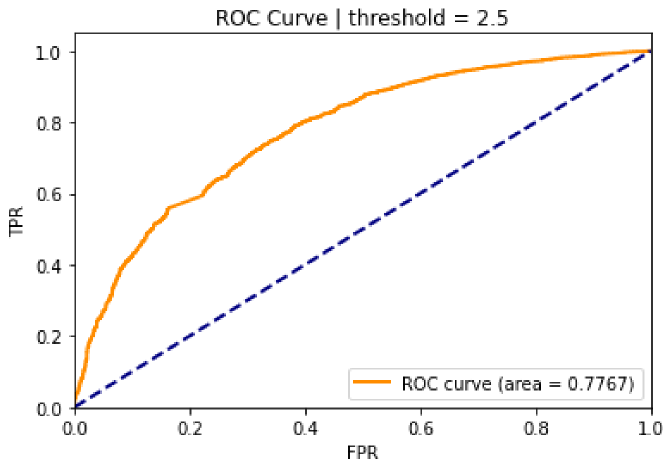
```
from sklearn.metrics import roc_curve, auc
```

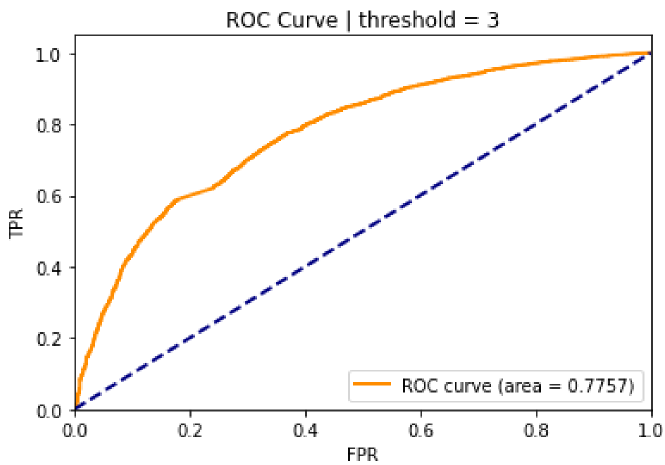
In [26]:

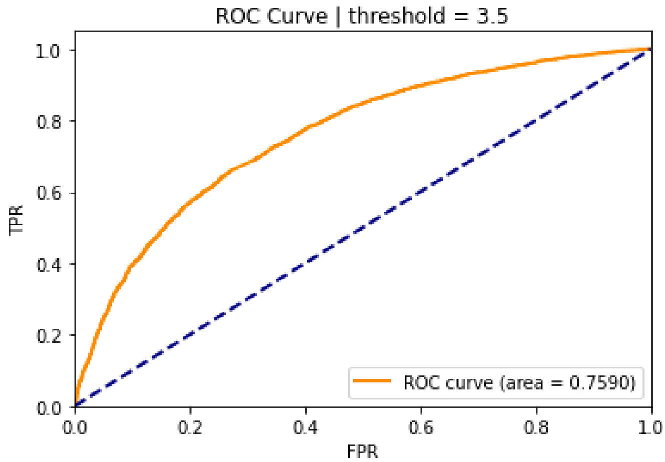
```
def plot_roc_curve(model, data, threshold, title=None):
    train, test = train_test_split(data, train_size=0.9, test_size=0.1)
    y_pred = model.fit(train).test(test)
    y_true = [0 if y.r_ui < threshold else 1 for y in y_pred]
    y_pred = [y.est for y in y_pred]
    fpr, tpr, _ = roc_curve(y_true, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',lw=lw, label='ROC curve (area = %0.4f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('FPR')
    plt.ylabel('TPR')
    if title:
        plt.title(title)
    plt.legend(loc="lower right")
    plt.show()
```

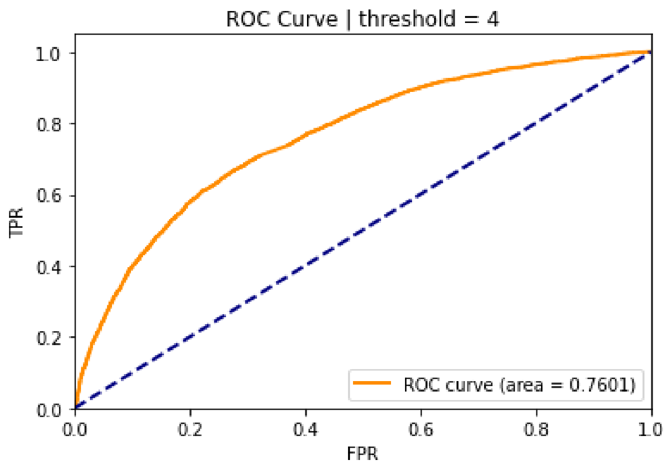
In [43…

```
best_k = 16
nmf = NMF(n_factors=best_k)

thresholds = [2.5, 3, 3.5, 4]
for t in thresholds:
    plot_roc_curve(nmf, data, t, "ROC Curve | threshold = " + str(t))
```









Q23

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?

ANSWER

Each latent factor (LF) seems to correspond to a relatively small collection of genres, but they aren't particularly distinctive. For each LF, we plotted the distribution of movie genre assignments to see which were the most prevalent. The lowest number of unique genres was 7 and the highest was 14. The most number of movies belonging to the same genre for a given LF was 7 out of 10, and 4 out of 10 for the lowest.

As for the connection between the LFs and the genres, there does seem to be a weak connection between them. For example LF={19} shows a strong presence of thriller, LF={1,2,8,12} for comedy, but many others show the dominance of drama in the genre assignments.

In [9]:

```
def get_latent_factor_details(genres, k):
    genres, counts = np.unique(genres, return_counts=True)
    print("unique genres: {}".format(len(genres)))
    print("total genres:  {}".format(counts.sum()))

    m_df = pd.DataFrame([genres, counts]).T
    m_df.rename(columns={0:'genres',1:'counts'}, inplace=True)

    g=sns.barplot(
```

```
        data=m_df,
        x=m_df.index,
        y='counts',
        palette='flare'
    )
    g.set_xticklabels(m_df.genres, rotation=90)
    g.set_title("Distribution of Genres for k=" + str(k))
    plt.show()
```

In [30…

```
movies_df = pd.read_csv('movies.csv')
train, test = train_test_split(data, train_size=0.9, test_size=0.1)
```

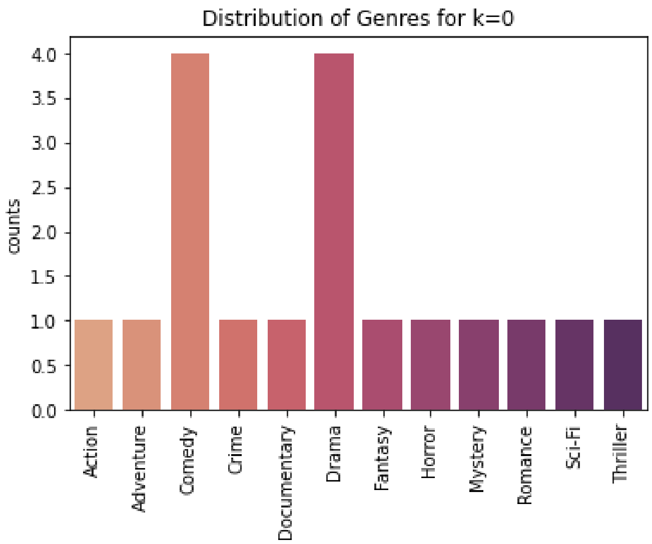
In [31…

```
nmf = NMF(n_factors=20)
nmf.fit(train).test(test)
U, V = nmf.pu, nmf.qi
```

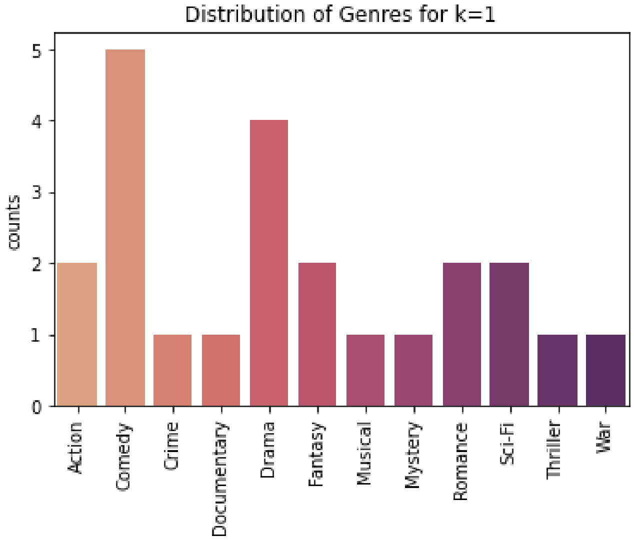
In [41…

```
for k in np.arange(0, 20):
    print('latent_factor: {}'.format(k))
    movies = [(n,j) for n,j in enumerate(V[:,k])]
    movies.sort(key=lambda x: x[1], reverse=True)
    genres = []
    for m in movies[:10]:
        genres.extend(movies_df['genres'].iloc[m[0]].split("|"))
    get_latent_factor_details(genres, k)
```

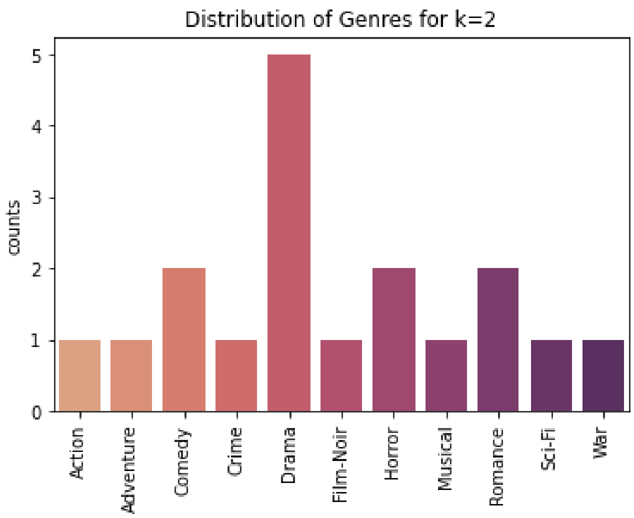
```
latent_factor: 0
unique genres: 12
total genres:  18
```



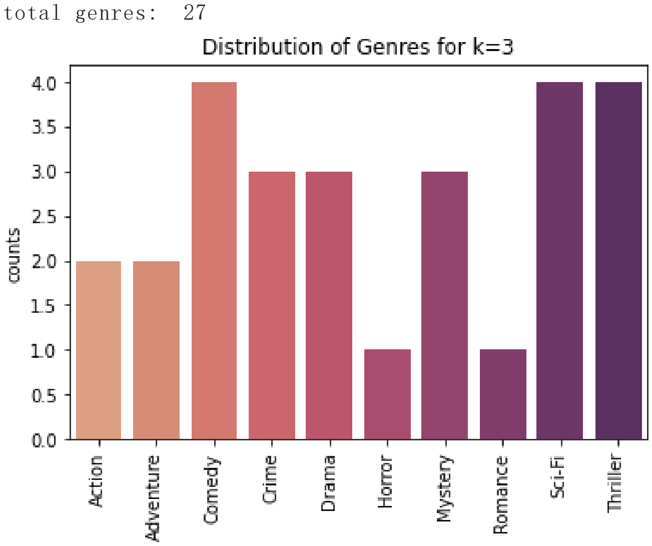

```
latent_factor: 1
unique genres: 12
total genres:  23
```



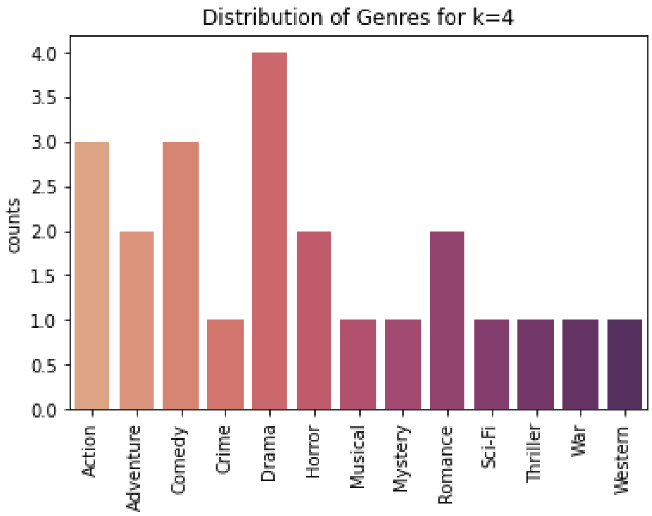

```
latent_factor: 2
unique genres: 11
total genres:  18
```



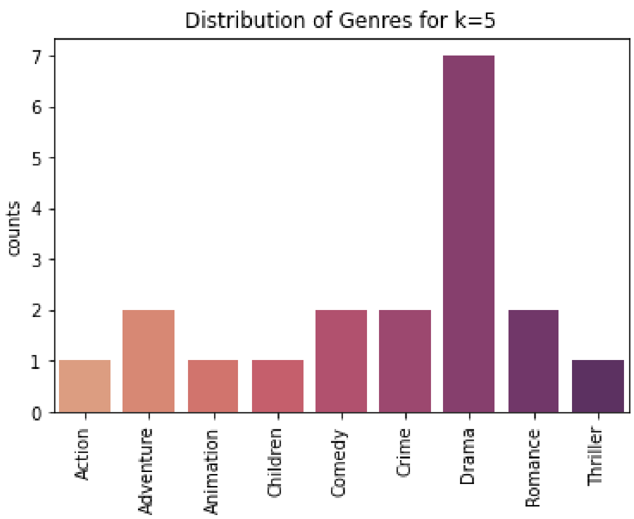

```
latent_factor: 3
unique genres: 10
total genres:  27
```



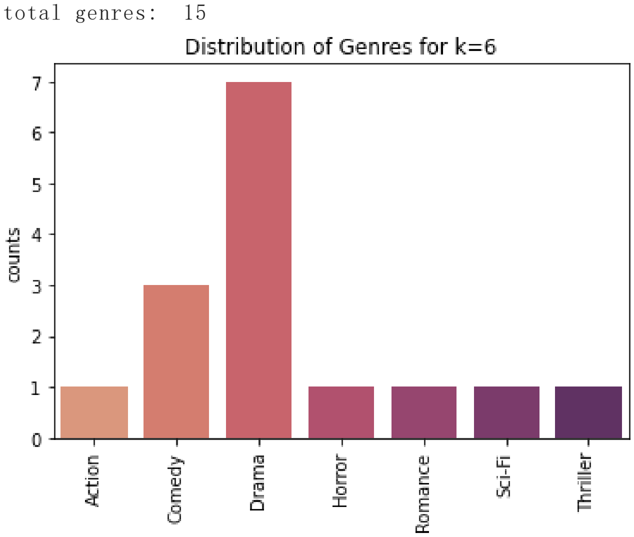

```
latent_factor: 4
unique genres: 13
total genres:  23
```



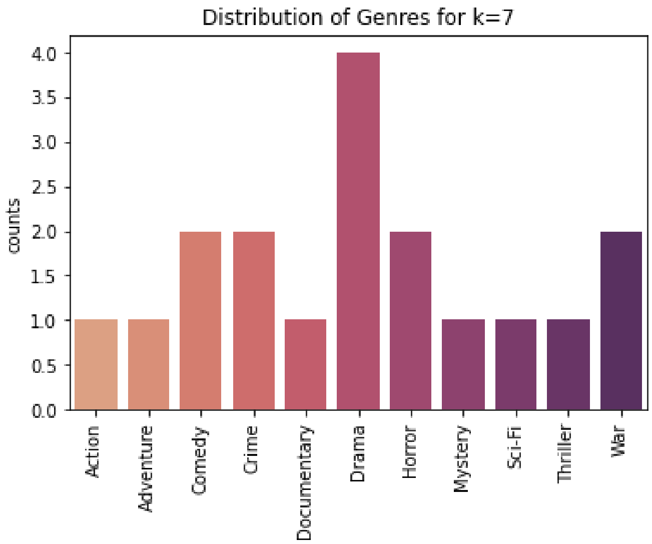

```
latent_factor: 5
unique genres: 9
total genres:  19
```



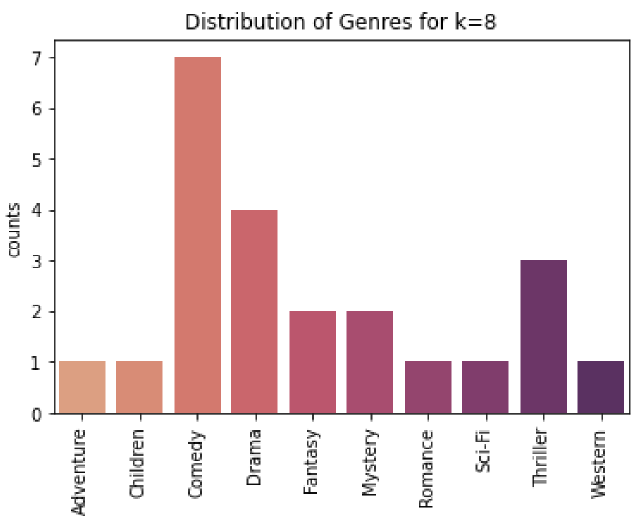

```
latent_factor: 6
unique genres: 7
total genres:  15
```



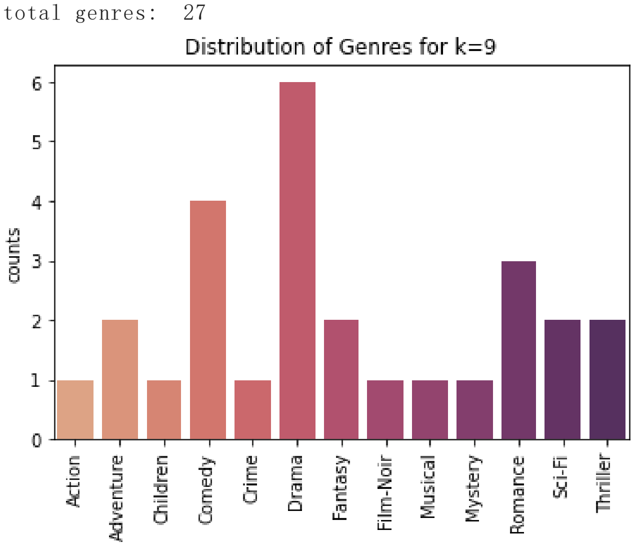

```
latent_factor: 7
unique genres: 11
total genres:  18
```



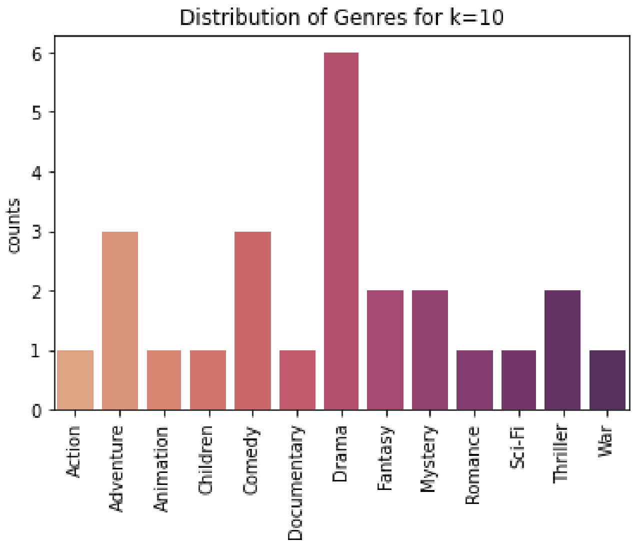

```
latent_factor: 8
unique genres: 10
total genres:  23
```



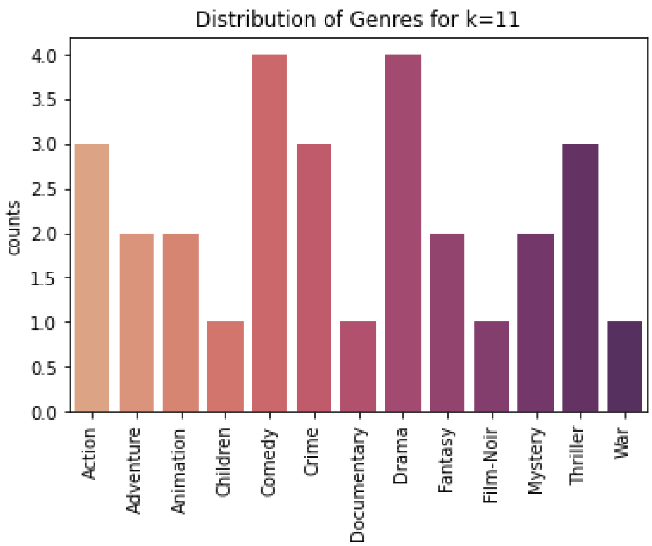

```
latent_factor: 9
unique genres: 13
total genres:  27
```



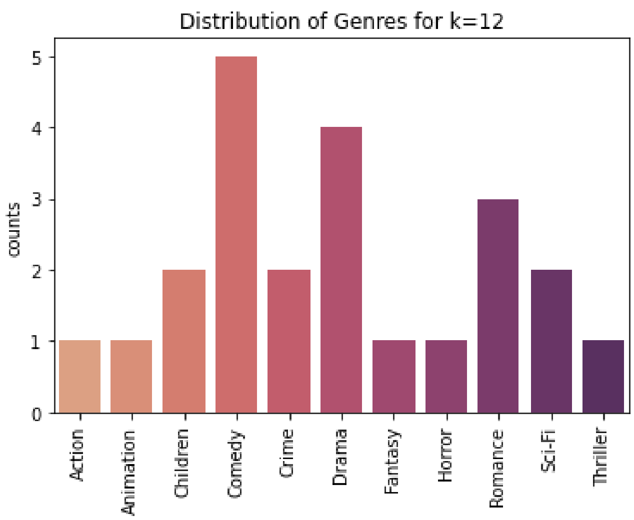

```
latent_factor: 10
unique genres: 13
total genres:  25
```



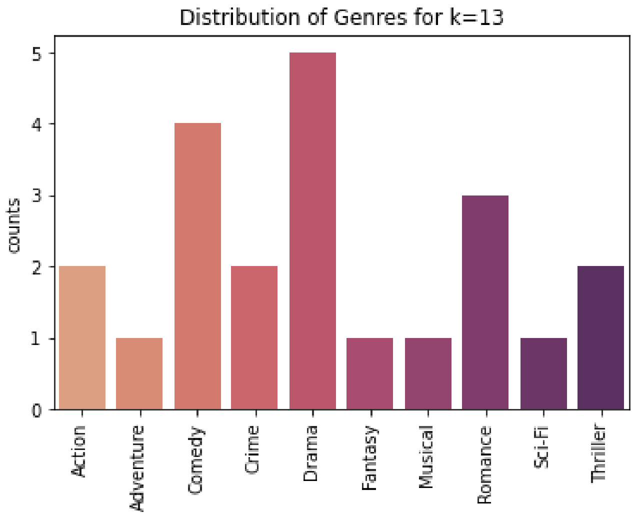

```
latent_factor: 11
unique genres: 13
total genres:  29
```



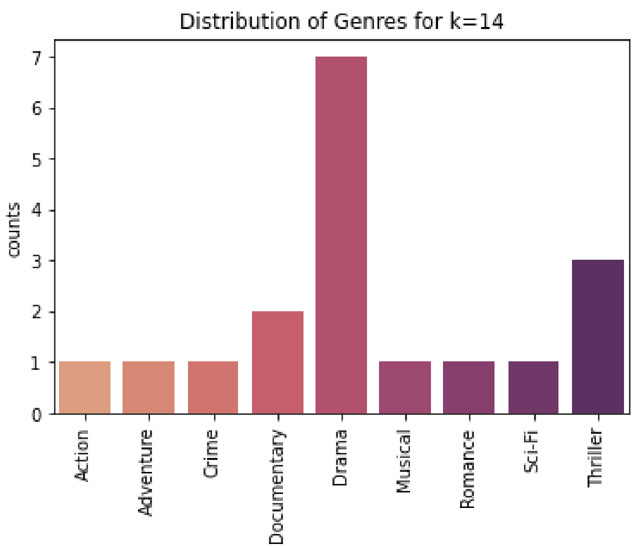

```
latent_factor: 12
unique genres: 11
total genres:  23
```



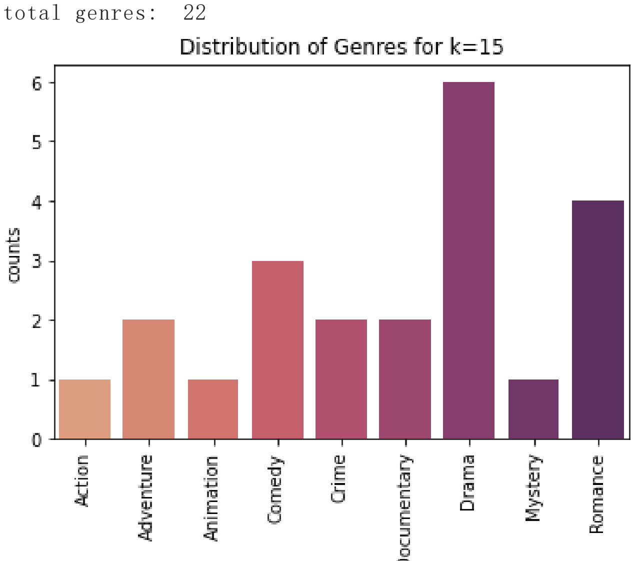

```
latent_factor: 13
unique genres: 10
total genres:  22
```



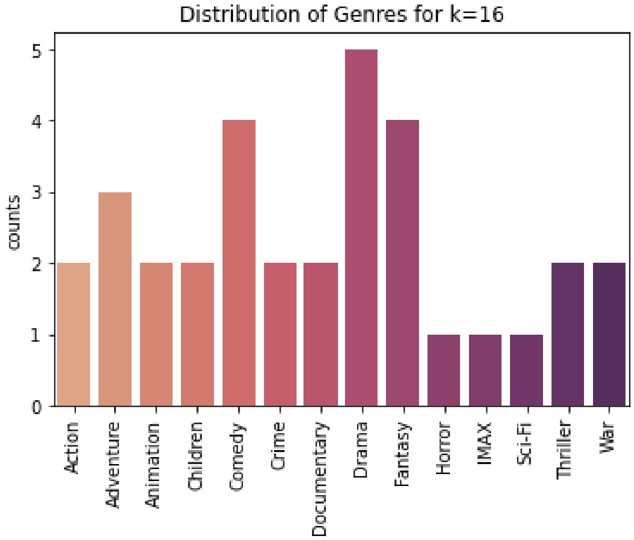

```
latent_factor: 14
unique genres: 9
total genres:  18
```



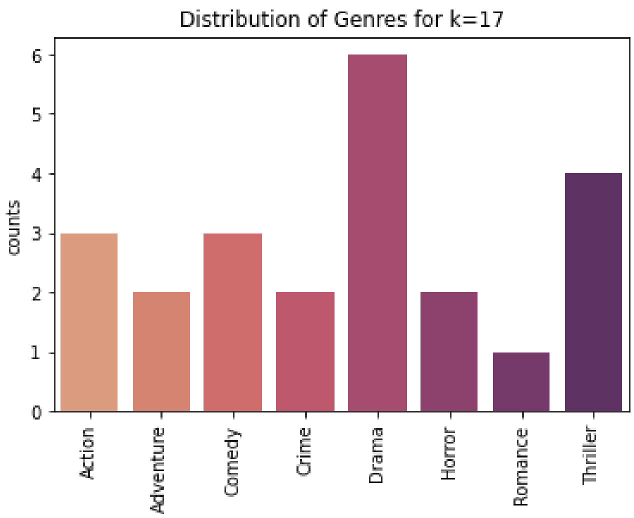

```
latent_factor: 15
unique genres: 9
total genres:  22
```



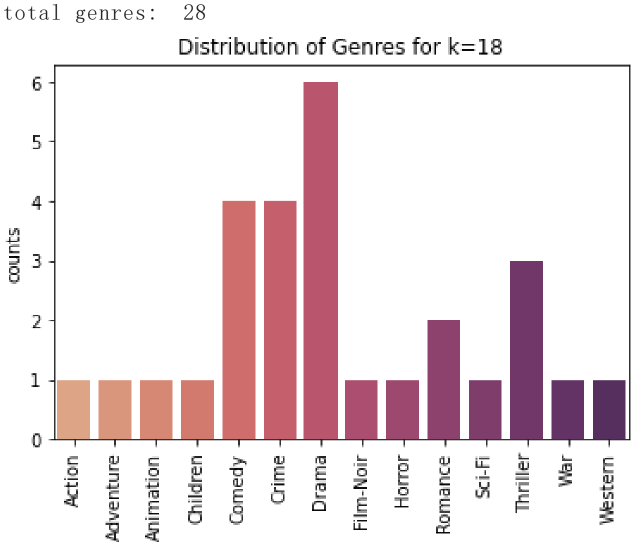

```
latent_factor: 16
unique genres: 14
total genres:  33
```



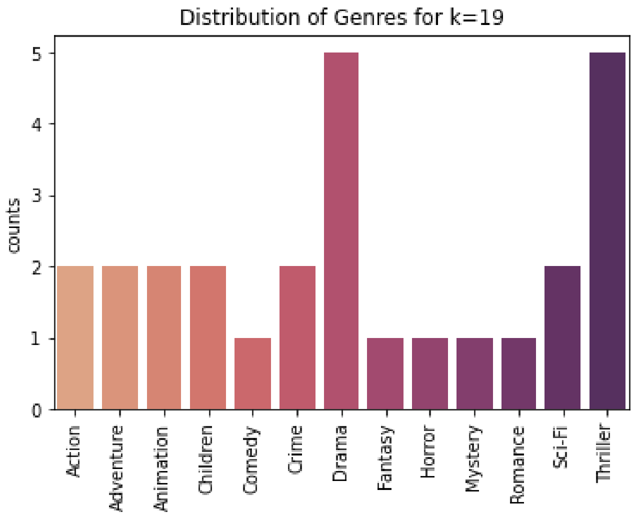

```
latent_factor: 17
unique genres: 8
total genres:  23
```




```
latent_factor: 18
unique genres: 14
total genres:  28
```




```
latent_factor: 19
unique genres: 13
total genres:  27
```



Q24

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

In [4]:

```
from surprise.prediction_algorithms.matrix_factorization import SVD
```

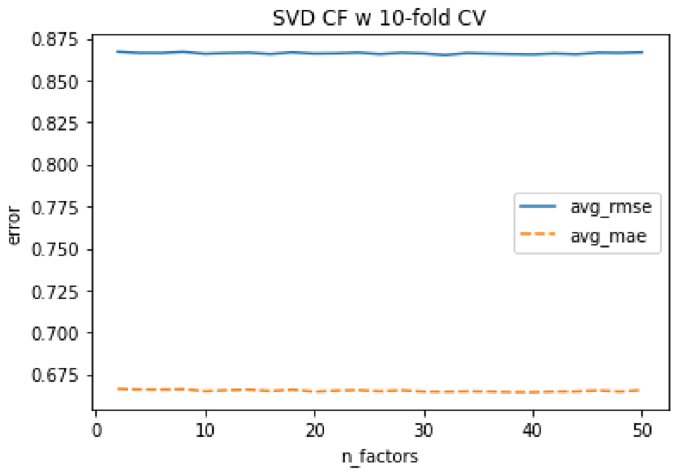
In [14]:

```
ks = np.arange(2, 51, 2)

results = []
for k in ks:
    print('Running with {} factors...'.format(k))
    perf = cross_validate(SVD(n_factors=k, biased=True), data, cv=10)
    results.append([k, perf['test_rmse'].mean(), perf['test_mae'].mean()])

df = pd.DataFrame(results, columns=['ks', 'avg_rmse', 'avg_mae']).set_index('ks')
```

```
Running with 2 factors...
Running with 4 factors...
Running with 6 factors...
Running with 8 factors...
Running with 10 factors...
Running with 12 factors...
Running with 14 factors...
Running with 16 factors...
Running with 18 factors...
Running with 20 factors...
Running with 22 factors...
Running with 24 factors...
Running with 26 factors...
Running with 28 factors...
Running with 30 factors...
Running with 32 factors...
Running with 34 factors...
Running with 36 factors...
Running with 38 factors...
Running with 40 factors...
Running with 42 factors...
Running with 44 factors...
Running with 46 factors...
Running with 48 factors...
Running with 50 factors...
```

In [15]:

```
g = sns.lineplot(data=df)
g.set_xlabel('n_factors')
g.set_ylabel('error')
g.set_title('SVD CF w 10-fold CV')
plt.show()
```



In [16]:

```
df.sort_values('avg_rmse').head(1)
```

Out[16]:

ks	avg_rmse	avg_mae
32	0.865094	0.664705

In [17]:

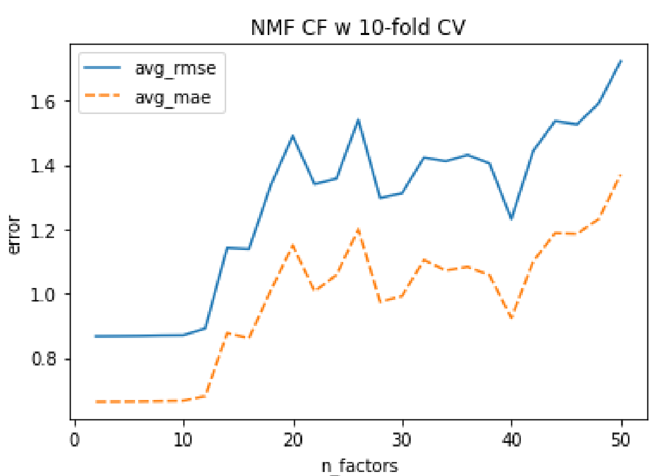
```
df.sort_values('avg_mae').head(1)
```

Out[17]:

ks	avg_rmse	avg_mae
40	0.865446	0.664435

In [41…

```
g = sns.lineplot(data=df)
g.set_xlabel('n_factors')
g.set_ylabel('error')
g.set_title('NMF CF w 10-fold CV')
plt.show()
```



In [41…

```
df.sort_values('avg_rmse').head(1)
```

Out[417]:

ks	avg_rmse	avg_mae
2	0.867691	0.664269

In [41…

```
df.sort_values('avg_mae').head(1)
```

Out[418]:

ks	avg_rmse	avg_mae
2	0.867691	0.664269

Q25

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

ANSWER

SVD w Bias

```
Minimum Average RMSE: 0.865094 @ k=32
Minimum Average MAE:  0.664435 @ k=40
```

NMF w Bias

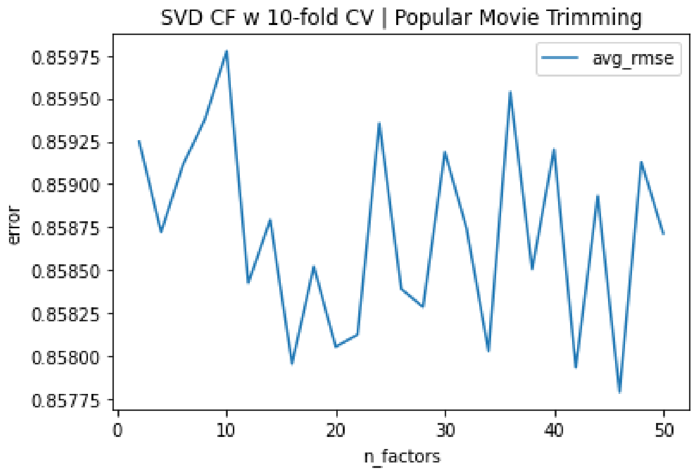
```
Minimum Average RMSE: 0.867691 @ k=2
Minimum Average MAE:  0.664269 @ k=2
```

Q26

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

In [22]:

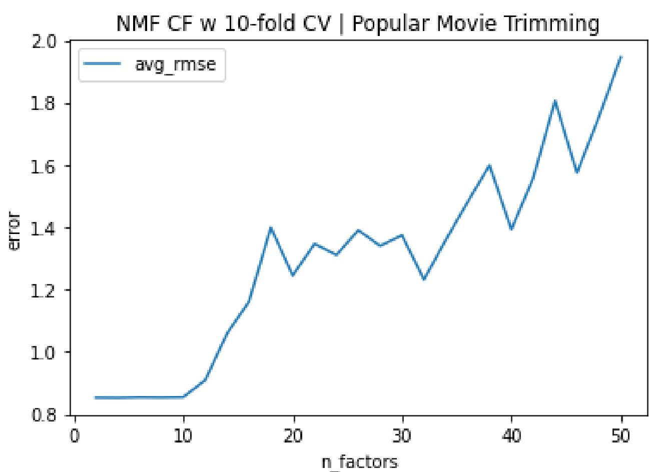
```
MF_plot(MF_type='SVD', trim="Popular", biased=True)
```




```
    avg_rmse
ks
46   0.85779
```

In [43…

```
MF_plot(MF_type='NMF', trim="Popular", biased=True)
```



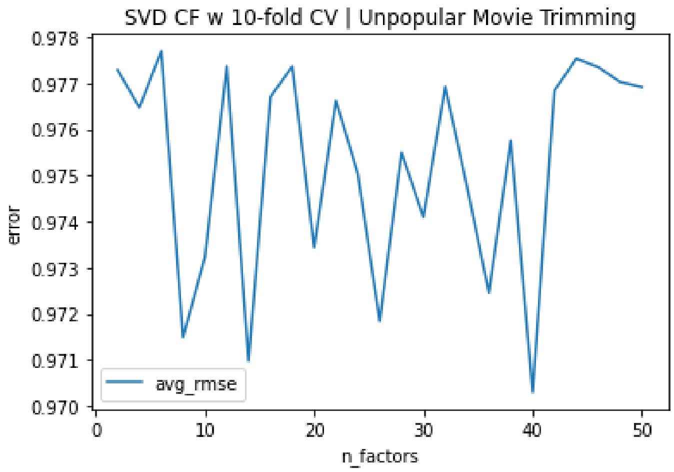

```
    avg_rmse
ks
4   0.853147
```

Q27

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

In [23]:

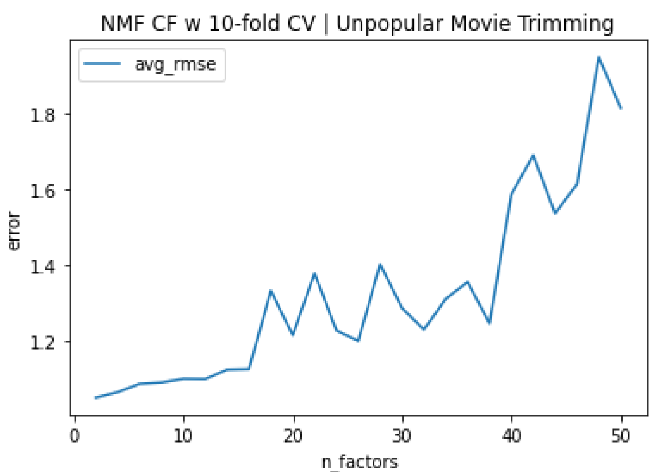
```
MF_plot(MF_type='SVD', trim="Unpopular", biased=True)
```




```
     avg_rmse
ks
40   0.970299
```

In [43…

```
MF_plot(MF_type='NMF', trim="Unpopular", biased=True)
```



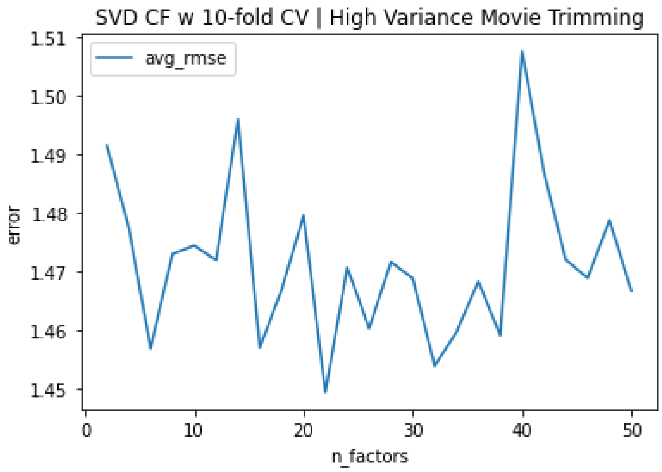

```
     avg_rmse
ks
2    1.052906
```

Q28

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

In [24]:

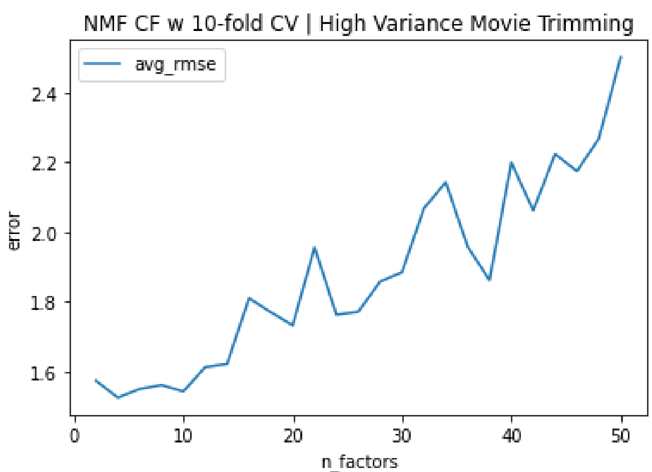
```
MF_plot(MF_type='SVD', trim="High Variance", biased=True)
```




```
    avg_rmse
ks
22  1.449506
```

In [43…

```
MF_plot(MF_type='NMF', trim="High Variance", biased=True)
```



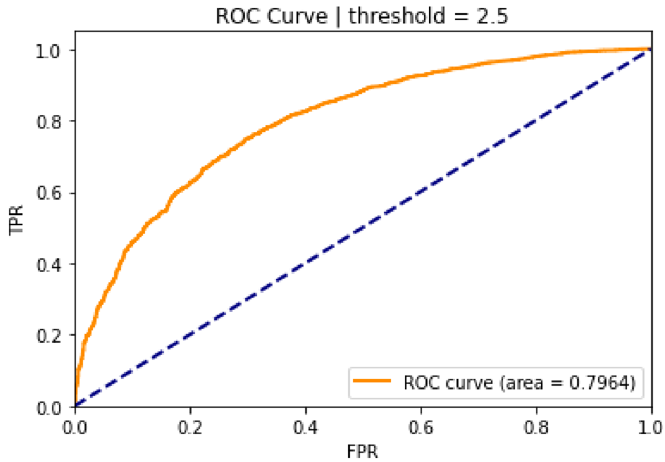

```
    avg_rmse
ks
4   1.524563
```

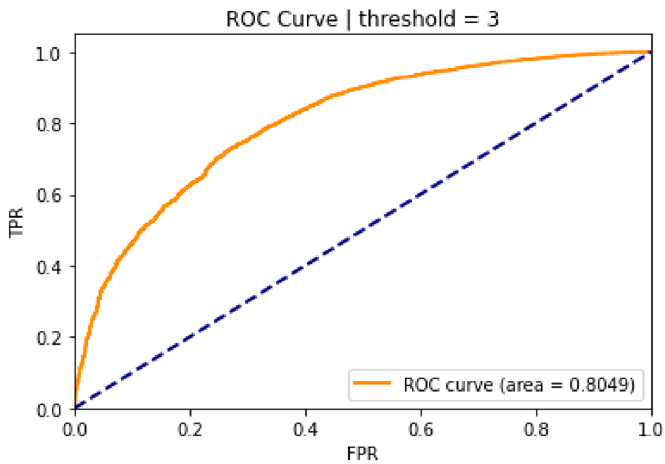
Q29

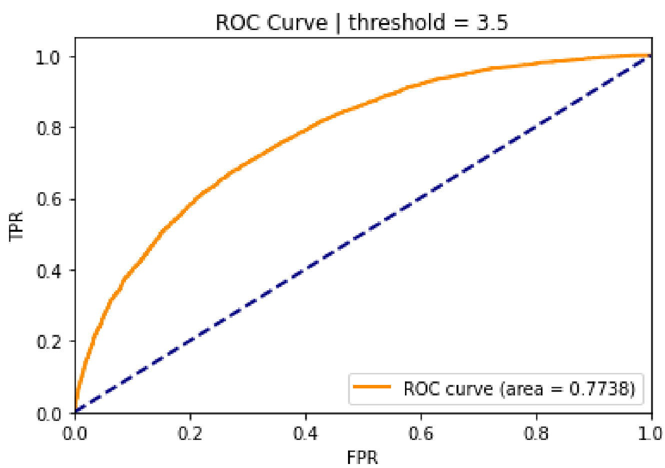
Plot the ROC curves for the MF with bias collaborative filter designed in question 24 for threshold values [2.5, 3, 3.5, 4]. For the ROC plotting use the optimal number of latent factors found in question 25. For each of the plots, also report the area under the curve (AUC) value.

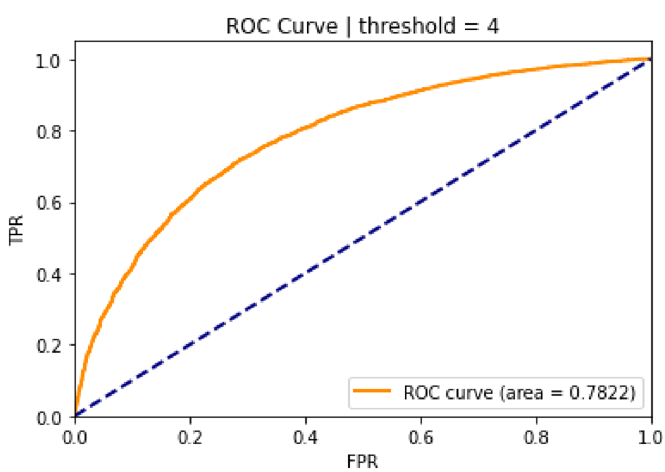
In [29]:

```
best_k = 32
svd = SVD(n_factors=best_k, biased=True)

thresholds = [2.5, 3, 3.5, 4]
for t in thresholds:
    plot_roc_curve(svd, data, t, "ROC Curve | threshold = " + str(t))
```


ROC Curve | threshold = 2.5
TPR
FPR
ROC curve (area = 0.7964)
0.0
0.2
0.4
0.6
0.8
1.0

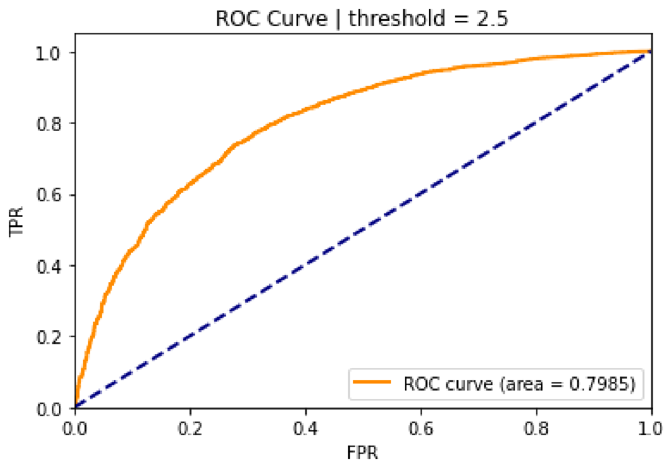

ROC Curve | threshold = 3
TPR
FPR
ROC curve (area = 0.8049)
0.0
0.2
0.4
0.6
0.8
1.0

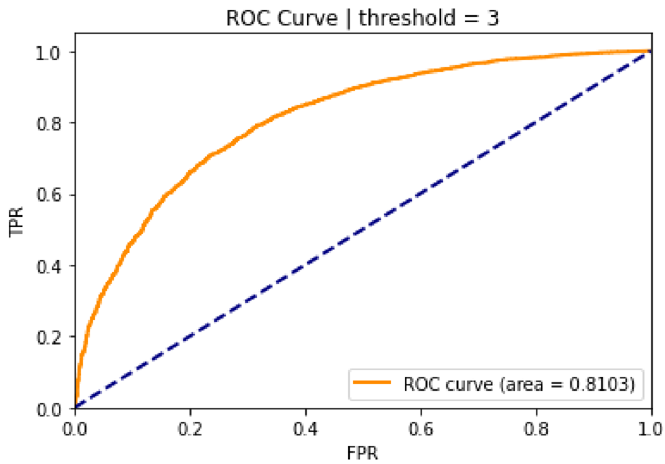

ROC Curve | threshold = 3.5
TPR
FPR
ROC curve (area = 0.7738)
0.0
0.2
0.4
0.6
0.8
1.0

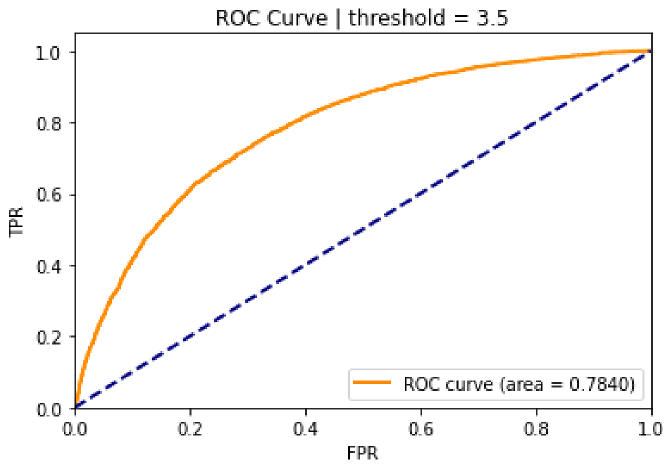

ROC Curve | threshold = 4
TPR
FPR
ROC curve (area = 0.7822)
0.0
0.2
0.4
0.6
0.8
1.0

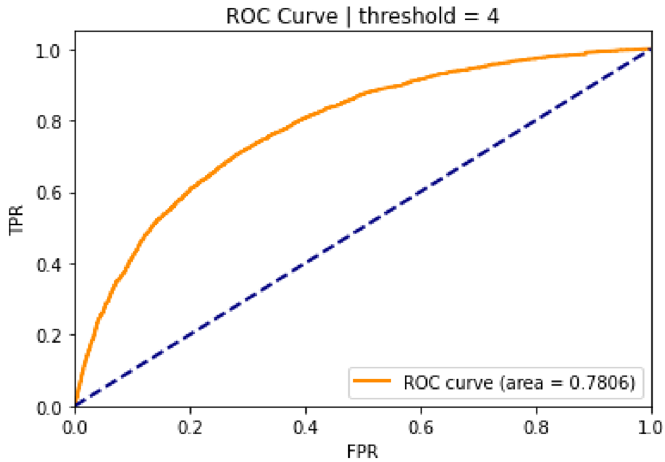
In [43…

```
best_k = 2
nmf = SVD(n_factors=best_k, biased=True)

thresholds = [2.5, 3, 3.5, 4]
for t in thresholds:
    plot_roc_curve(nmf, data, t, "ROC Curve | threshold = " + str(t))
```








ROC Curve | threshold = 4
TPR
FPR
ROC curve (area = 0.7806)
0.0
0.2
0.4
0.6
0.8
1.0

Include required packages

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from surprise import Dataset, Reader
from surprise.model_selection import KFold, train_test_split
from surprise.prediction_algorithms.knns import KNNWithMeans
from surprise.prediction_algorithms.matrix_factorization import NMF, SVD
from sklearn.metrics import mean_squared_error, roc_curve
```

Below is naive collaborative filter, and we report the average RMSE of 10 flod cross validation using test dataset. (Note: navie collaborative filter doesn't need training). The resulting RMSE score is 9.347089292911079.

In [2]:

```
file_path = "./ratings.csv"
reader = Reader(line_format="user item rating timestamp", sep=",",skip_lines=1)
dataset = Dataset.load_from_file(file_path, reader=reader)

naive_filter = {}
rating_count = {}
for user, _, rating, _ in dataset.raw_ratings:
    rating_count[user] = rating_count.get(user, 0) + 1
    naive_filter[user] = naive_filter.get(user, 0) + rating
for user in naive_filter.keys():
    naive_filter[user] = naive_filter[user]/rating_count[user]

RMSE = 0
kf = KFold(n_splits=10)
for _, test_data in kf.split(dataset):
    pred = []
    valid = []
    for data in test_data:
        pred.append(naive_filter[data[0]])
        valid.append(data[2])
    RMSE += np.sqrt(mean_squared_error(valid, pred))
RMSE
```

Out[2]: 9.347075046573224

We apply the naive collaborative filter on popular movie trimmed test set. Specifically, we ignore any moive that have less or equal to 2 ratings and report the average RMSE of 10 flod cross validation using test dataset. The resulting RMSE score is 9.323127210045904.

In [3]:

```
movie_rating_count = {}
for _, movie, _, _ in dataset.raw_ratings:
    movie_rating_count[movie] = movie_rating_count.get(movie, 0) + 1


RMSE = 0
kf = KFold(n_splits=10)
for _, test_data in kf.split(dataset):
    trimmed_data = []
    for user,movie,rating in test_data:
        if movie_rating_count[movie] > 2:
            trimmed_data.append((user,movie,rating))
    pred = []
    valid = []
    for data in trimmed_data:
        pred.append(naive_filter[data[0]])
        valid.append(data[2])
    RMSE += np.sqrt(mean_squared_error(valid, pred))
RMSE
```

Out[3]: 9.323194526590324

We apply the naive collaborative filter on unpopular movie trimmed test set. Specifically, we ignore any moive that have more than to 2 ratings and report the average RMSE of 10 flod cross validation using test dataset. The resulting RMSE score is 9.710962250099168.

In [4]:
```
movie_rating_count = {}
for _, movie, _, _ in dataset.raw_ratings:
    movie_rating_count[movie] = movie_rating_count.get(movie, 0) + 1


RMSE = 0
kf = KFold(n_splits=10)
for _, test_data in kf.split(dataset):
    trimmed_data = []
    for user,movie,rating in test_data:
        if movie_rating_count[movie] <= 2:
            trimmed_data.append((user,movie,rating))
    pred = []
    valid = []
    for data in trimmed_data:
        pred.append(naive_filter[data[0]])
        valid.append(data[2])
    RMSE += np.sqrt(mean_squared_error(valid, pred))
RMSE
```

Out[4]: 9.708386728672012

We apply the naive collaborative filter on high variance movie trimmed test set. Specifically, we only consider movie with at least 5 ratings and at least 2 points rating variance. Same as before, we report the average RMSE of 10 flod cross validation using test dataset. The resulting RMSE score is 9.350126702356985.

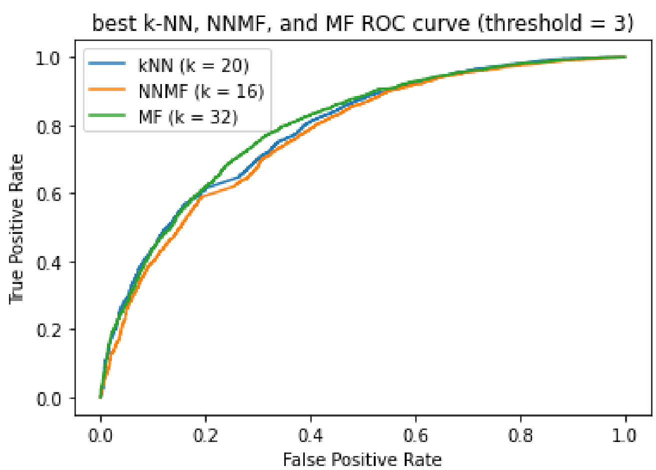
In [5]:
```
movie_rating_count = {}
movie_rating_minmax = {}
for _, movie, rating, _ in dataset.raw_ratings:
    movie_rating_count[movie] = movie_rating_count.get(movie, 0) + 1
    min_rating, max_rating = movie_rating_minmax.get(movie, (5, 0.5))
    min_rating = min(min_rating, rating)
    max_rating = max(max_rating, rating)
    movie_rating_minmax[movie] = (min_rating, max_rating)


RMSE = 0
kf = KFold(n_splits=10)
for _, test_data in kf.split(dataset):
    trimmed_data = []
    for user,movie,rating in test_data:
        if movie_rating_count[movie] >= 5 and movie_rating_minmax[movie][1] - movie_rating_minmax[movie][
            trimmed_data.append((user,movie,rating))
    pred = []
    valid = []
    for data in trimmed_data:
        pred.append(naive_filter[data[0]])
        valid.append(data[2])
    RMSE += np.sqrt(mean_squared_error(valid, pred))
RMSE
```

Out[5]: 9.350079655577108

We show the ROC curve using of k-NN (k=20), NNMF(k=16), and MF(k=32) with bias based collaborative filters (threshold set to 3). As we can see in the figure below, MF with bias based collaborative filter slightly outperform k-NN and NNMF. It has the largest area under ROC curve, which means it produce better movie rating predictions.

In [6]:
```
def plot_ROC_curve(pred, label):
    y_real = []
    y_pred = []
    for i, p in enumerate(pred):
```

```
        y_pred.append(pred[i].est)
        y_real.append(int(test_data[i][2] >= 3))
    fpr, tpr, _ = roc_curve(y_real, y_pred)
    plt.plot(fpr, tpr, label=label)

train_data, test_data = train_test_split(dataset, test_size=.1)

pred = KNNWithMeans(k=20,sim_options={"name":"pearson"},verbose=False).fit(train_data).test(test_data)
plot_ROC_curve(pred, "kNN (k = 20)")

pred = NMF(n_factors=16,verbose=False).fit(train_data).test(test_data)
plot_ROC_curve(pred, "NNMF (k = 16)")

pred = SVD(n_factors=32,biased=True,verbose=False).fit(train_data).test(test_data)
plot_ROC_curve(pred, "MF (k = 32)")

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("best k-NN, NNMF, and MF ROC curve (threshold = 3)")
plt.legend(loc="best")
plt.show()
```



Precision is the fraction of liked and recommended items over the whole recommendation Recall is the fraction of liked and recommended items over everything liked.

We use KNN to estimate movie ratings and determine which movies to recommend to users. Below are the graphs showing the relations between percision, recall and t of our recommendation.

In [7]:

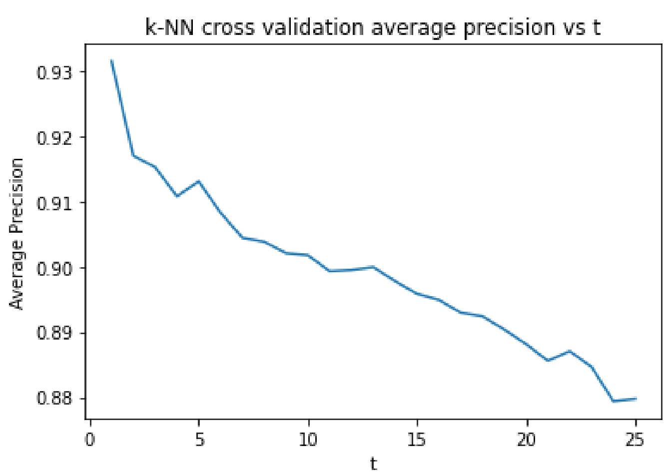
```
ts = [t for t in range(1, 26)]
def precision_recall(model):
    precision = []
    recall = []
    kf = KFold(n_splits=10)
    for t in ts:
        precision_sum_t = 0
        recall_sum_t = 0
        for train_data, test_data in kf.split(dataset):
            G = {}
            user_count = {}
            for user,movie,rating in test_data:
                if rating >= 3:
                    G[user] = G.get(user, set())
                    G[user].add(movie)
                user_count[user] = user_count.get(user, 0) + 1
            trimmed_data = []
            for user,movie,rating in test_data:
                if user_count[user] >= t and user in G:
                    trimmed_data.append((user,movie,rating))
            pred = model.fit(train_data).test(trimmed_data)
            user_recommendation = {}
            for user, movie, _, pred_rating, _ in pred:
```

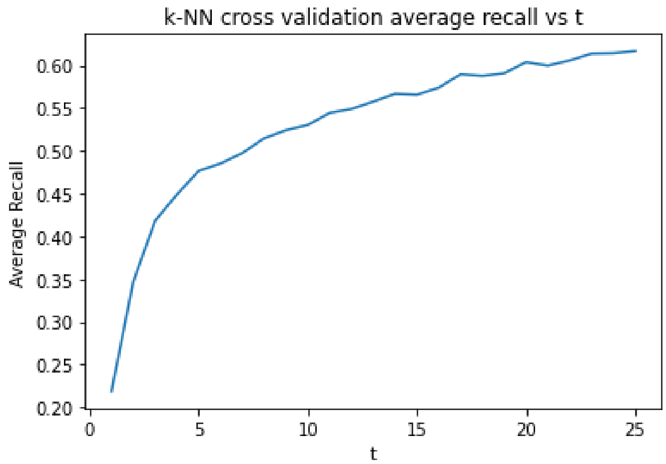
```
                user_recommendation[user] = user_recommendation.get(user, []) + [(pred_rating, movie)]
            precision_sum = 0
            recall_sum = 0
            for user in user_recommendation.keys():
                sorted_recommendation = sorted(user_recommendation[user],reverse=True)
                S_t = set(list(map(lambda x: x[1], sorted_recommendation[:t])))
                precision_sum += len(S_t.intersection(G[user]))/float(len(S_t))
                recall_sum += len(S_t.intersection(G[user]))/float(len(G[user]))
            precision_sum_t += precision_sum/len(user_recommendation)
            recall_sum_t += recall_sum/len(user_recommendation)
        precision.append(precision_sum_t/10)
        recall.append(recall_sum_t/10)
    return precision, recall

def plot_precision_recall(precision, recall, tital):
    plt.plot(ts, precision)
    plt.xlabel("t")
    plt.ylabel("Average Precision")
    plt.title(tital + " cross validation average precision vs t")
    plt.show()

    plt.plot(ts, recall)
    plt.xlabel("t")
    plt.ylabel("Average Recall")
    plt.title(tital + " cross validation average recall vs t")
    plt.show()

    plt.plot(recall,precision)
    plt.xlabel("Average Recall")
    plt.ylabel("Average Precision")
    plt.title(tital + " cross validation average precision vs average recall")
    plt.show()
    return
```

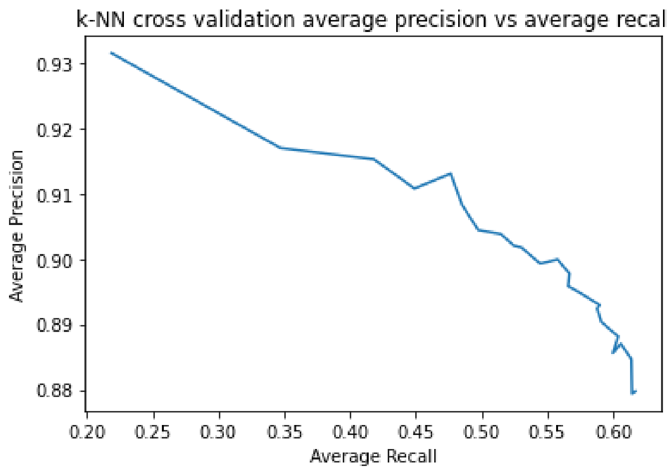
As we can see, precision and t have an negative correlation, which means precision score gets lower as we increase t; recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger); precision and recall have negative correlation, which means precision score is lower when recall score is higher.

In [8]:

```
knn_precision, knn_recall = precision_recall(KNNWithMeans(k=20,sim_options={"name":"pearson"},verbose=Fal
plot_precision_recall(knn_precision, knn_recall, "k-NN")
```



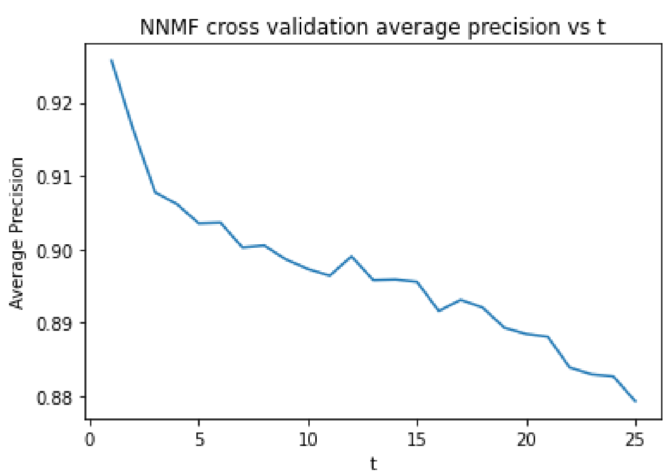


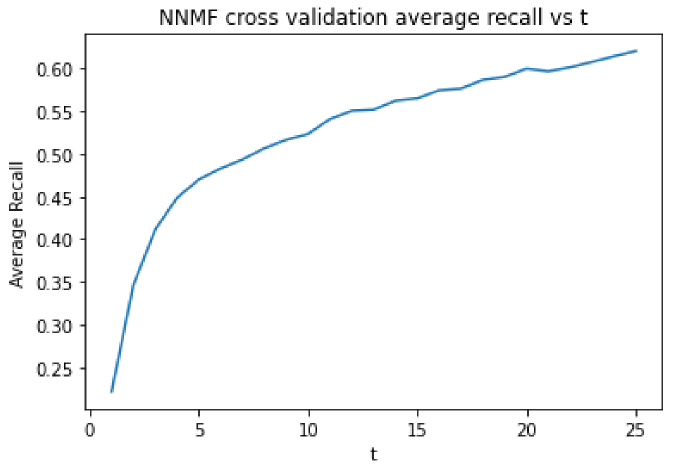


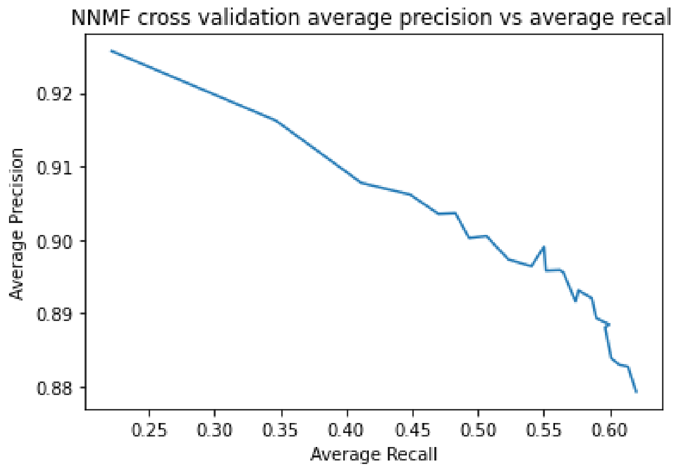
Similar to KNN result, precision and t have an negative correlation, which means precision score gets lower as we increase t; recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger); precision and recall have negative correlation, which means precision score is lower when recall score is higher.

In [9]:

```
nnmf_precision, nnmf_recall = precision_recall(NMF(n_factors=16,verbose=False))
plot_precision_recall(nnmf_precision, nnmf_recall, "NNMF")
```



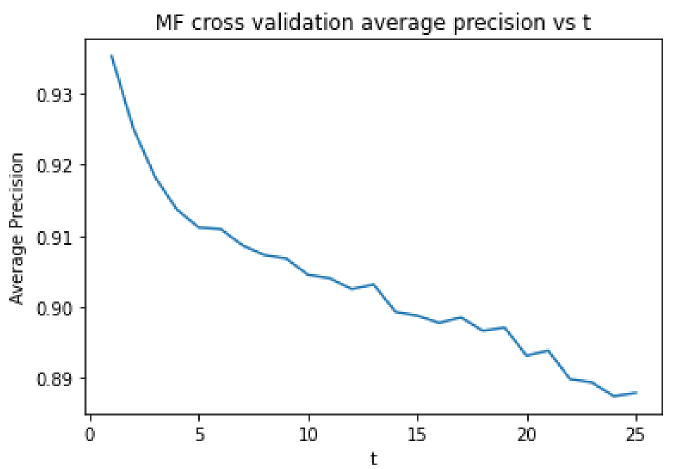


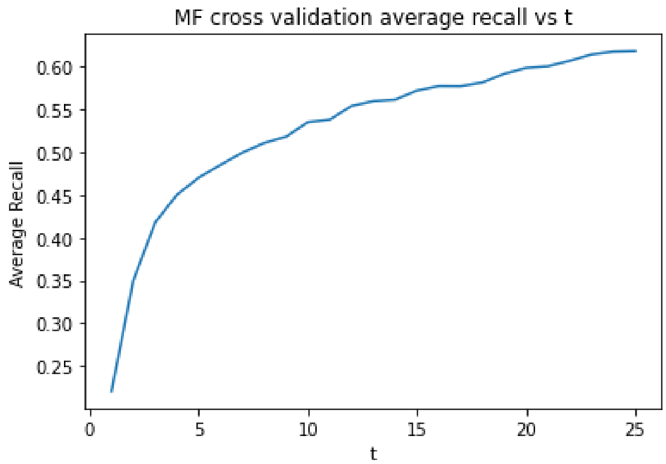


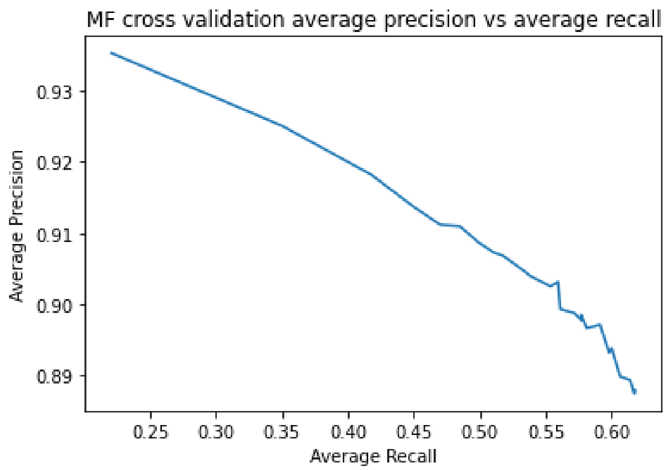
Similar to previous two result, precision and t have an negative correlation, which means precision score gets lower as we increase t; recall and t have an positive correlation, which means recall score gets higher as we increase t (the recall score increases slower as t gets larger); precision and recall have negative correlation, which means precision score is lower when recall score is higher.

In [10]:

```
mf_precision, mf_recall = precision_recall(SVD(n_factors=32,biased=True,verbose=False))
plot_precision_recall(mf_precision, mf_recall, "MF")
```







Here we show the precision-recall curve of k-NN, NNMF, and MF. As we can see, MF with bias-based collaborative filter's curve is slightly above the others', which means MF would have better precision and recall score in almost all given t value. Thus we can conclude that MF with bias-based collaborative filter allows us to produce the most relevant recommendations.

In [11]:

```
plt.plot(knn_recall, knn_precision, label = "knn")
plt.plot(nnmf_recall, nnmf_precision, label = "nnmf")
plt.plot(mf_recall, mf_precision, label = "mf")
plt.xlabel("Average Recall")
plt.ylabel("Average Precision")
plt.title("KNN, NNMF and MF cross validation average precision vs average recall")
plt.legend(loc="best")
plt.show()
```