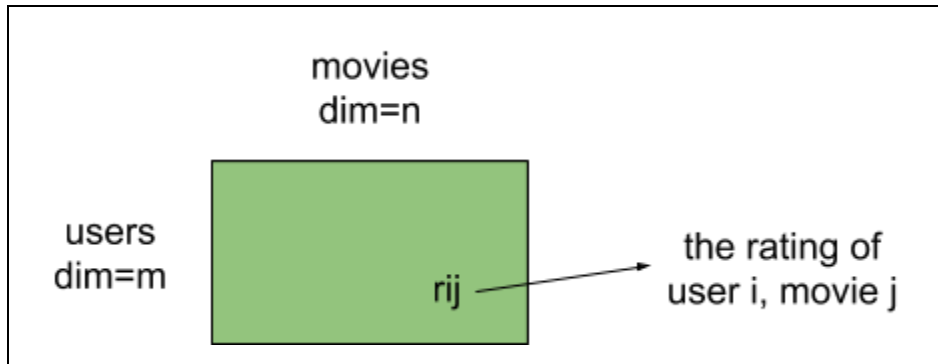


## Q1

First of all, because the original indices of the movies were not continuous, we re-indexed the movies. After this operation, the indices of the movies are integers from 1 to 9123. The indices of the users are from 1 to 671. We created a new rating list with the new indices, and saved it as “ratings\_new.csv”. That new list include the titles of the movies.

In order to do find the relationship among movies, users, and ratings. We constructed a rating matrix. As shown below.

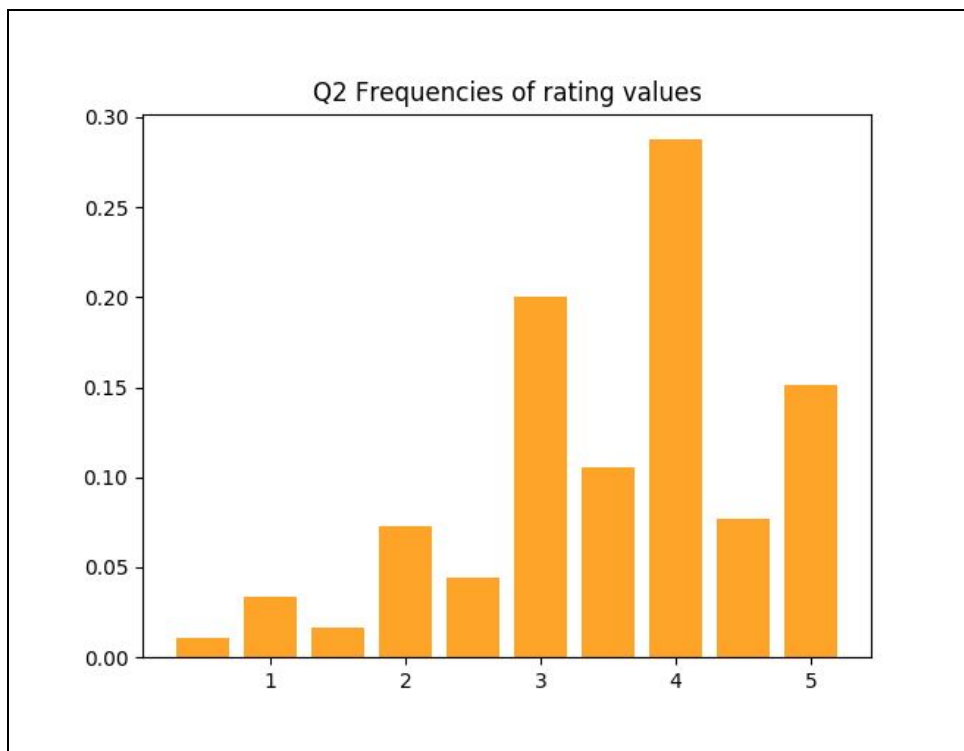


In the rating list, the minimum rating was 0.5 the maximum was 5.0. There were no 0.0, so if the entries of the matrix were zeros, that means there were no ratings. And then, we can calculate the sparsity of the matrix, which is the percentage of entries that have rating values. We counted the nonzero entries inside the matrix and found that the sparsity was 1.634%. This low percentage is reasonable, because no one will have time to rate thousands of movies.

## Q2

In this part, we count the fractions of each rating values. The values are [0.5,1,1.5,2,2.5,3,3.5,4,4.5,5]. After we counted the amounts of each rating, we normalized them into percentage.

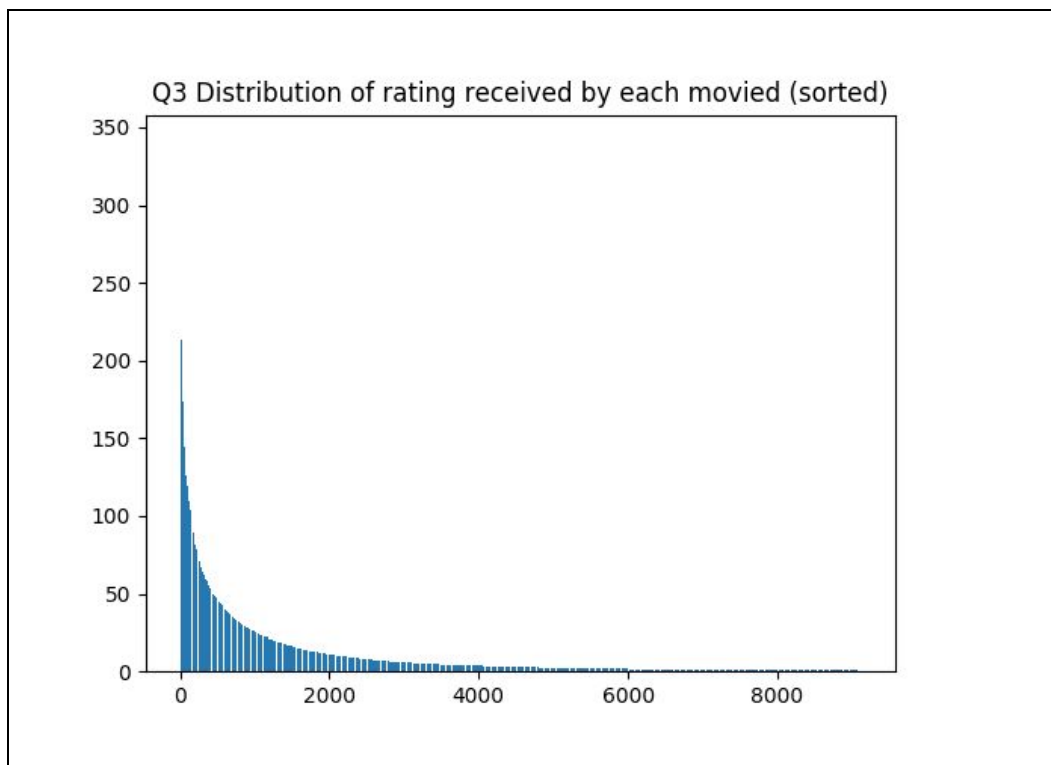
We plotted this into a histogram. As shown below. We can see that, the rating score of 4 was the most popular one. It is reasonable, because usually, people don't give full score unless the movies were really good. The scores of 1 and 2 were less. We can infer 2 possibilities. First, people watch good movies based on the titles and introductions. Second, if the movies were bad, they tend to shut down the TV and waste no time on rating. Furthermore, we can see that people tend to give integer scores.



### Q3

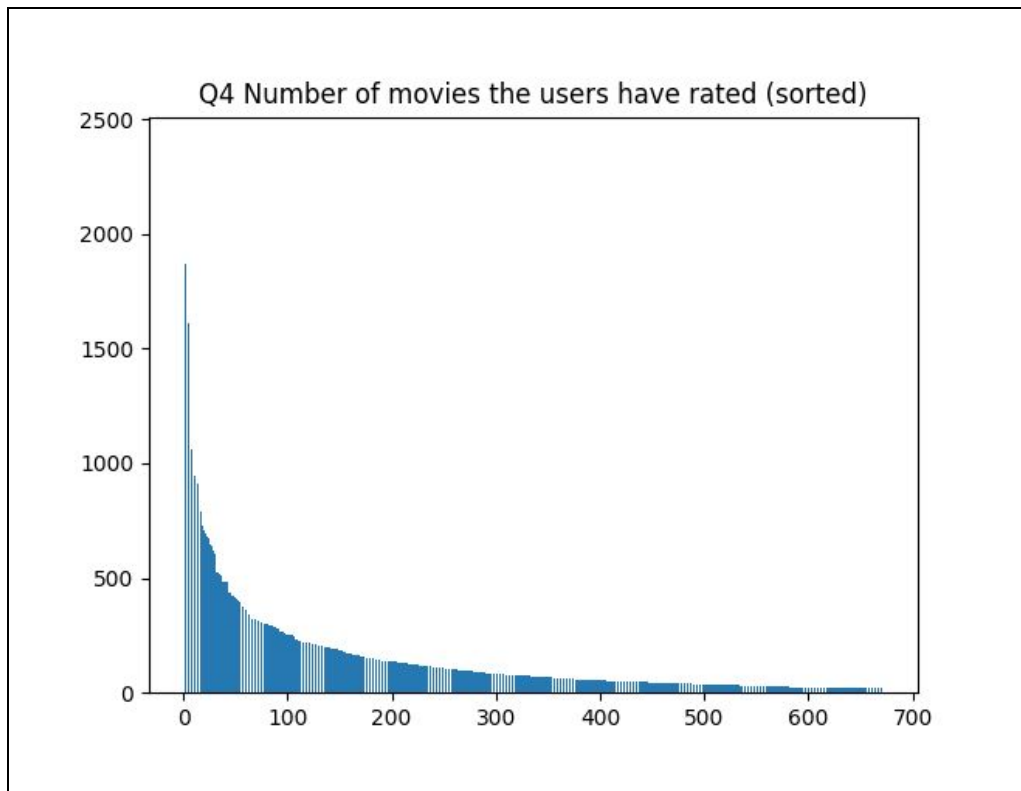
Now, we calculate the total ratings of each movies that they have received. The more the total ratings they have, the more popular. To do this, we just counted the nonzero entries of each columns. In order to see which movies were more popular, we sorted the sequence based on the total ratings of them. The sorted list with movie indices and rating amounts was stored in the matrix, "Q3\_index\_freq\_out". We list some of the popular movies in the table below.

movie index	number of ratings	movie title
322	341	Forrest Gump (1994)
267	324	Pulp Fiction (1994)
285	311	The Shawshank Redemption (1994)
526	304	Silence of the Lambs (1991)
233	291	Star Wars: Episode IV - A New Hope (1977)
.....		
5861	1 (rating=3)	Night and the City (1992)
.....		



### Q4

In this part, we want to see the numbers of movies the users have rated. To do this, we just counted the nonzero entries in each row. Again, we sorted the list with the numbers of rating. The sorted list with user indices and rating amounts was stored in the matrix, “Q4\_index\_freq\_out”. From the result, we can see that some users have rated more than a thousand movies. The top user, the user 547 has rated 2391 movies, who is a professional rater. Furthermore, we can see that people have at least rated 20 movies. The statistics is shown below.



## Q5

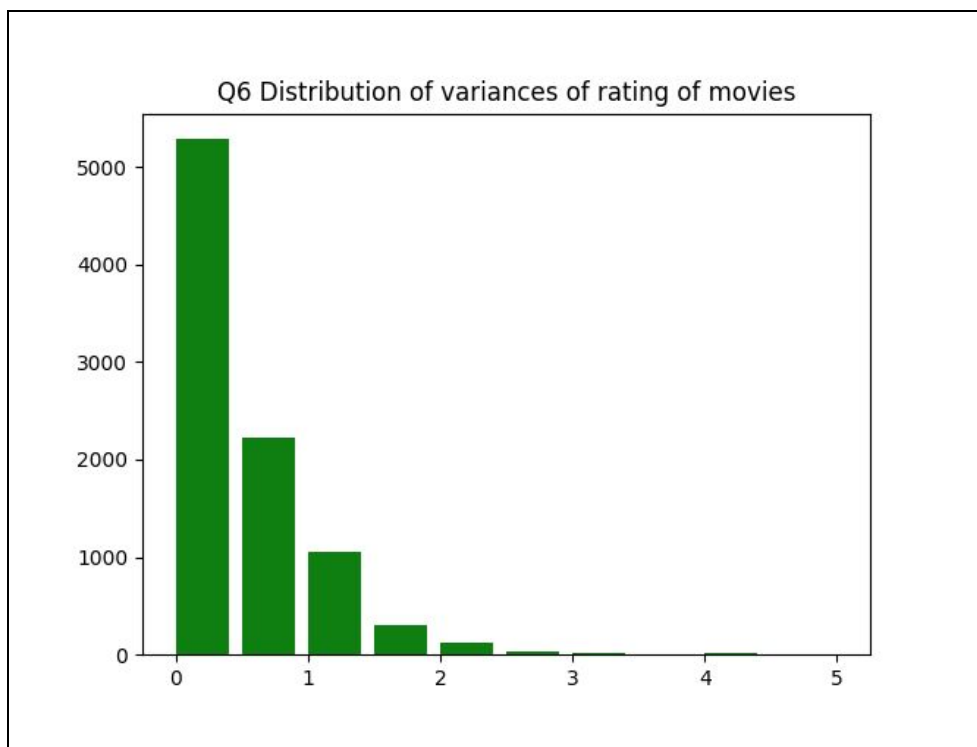
The feature in Q3 is straight forward. It lists the popular movies. The more a movie got the ratings, the more chance the movie will be liked by a random person. Note that, the recommendation process is for random person, not customized.

## Q6

In this part, we want to calculate the variances of rating values received by each movie. This is to calculate the variances of each columns but ignore zeros. Also, in Q3, we have found that 57 of the movies received no ratings. Therefore, we can not calculate their variances. The indices of the movies have no ratings are shown below.

```
3693 3732 3983 4047 4414 4523 4677 4756 4836 5073 5114 5368 5433 5530 5531 5700
6195 6268 6311 6328 6551 6594 6826 6830 7689 8062 8319 8355 8486 8573 8579 8580
8587 8621 8671 8674 8689 8752 8793 8794 8810 8820 8823 8869 8876 8881 8882 8885
8953 9021 9045 9075 9091 9109 9112 9113 9121
```

During the calculation of the variances of each movie, we deleted the zeros in each column. The result of the variance histogram is shown below. Note that there were only 9066 movies, which have a least one rating taken into consideration. We can see that most of the movies have small variances of rating, which mean people have similar tastes among the movies.



Q7

$$\mu_u = \frac{\sum_{k=1}^{length(I_u)} r_{uk}}{length(I_u)}$$

Q8

$I_u$  is the set of movies that user  $u$  has rated, while  $I_v$  is the set of movies that user  $v$  has rated.  $I_u \cap I_v$  is the set of movies that both user  $u$  and user  $v$  have rated. If  $I_u \cap I_v = \emptyset$ , that means the two users have no common rated movies.

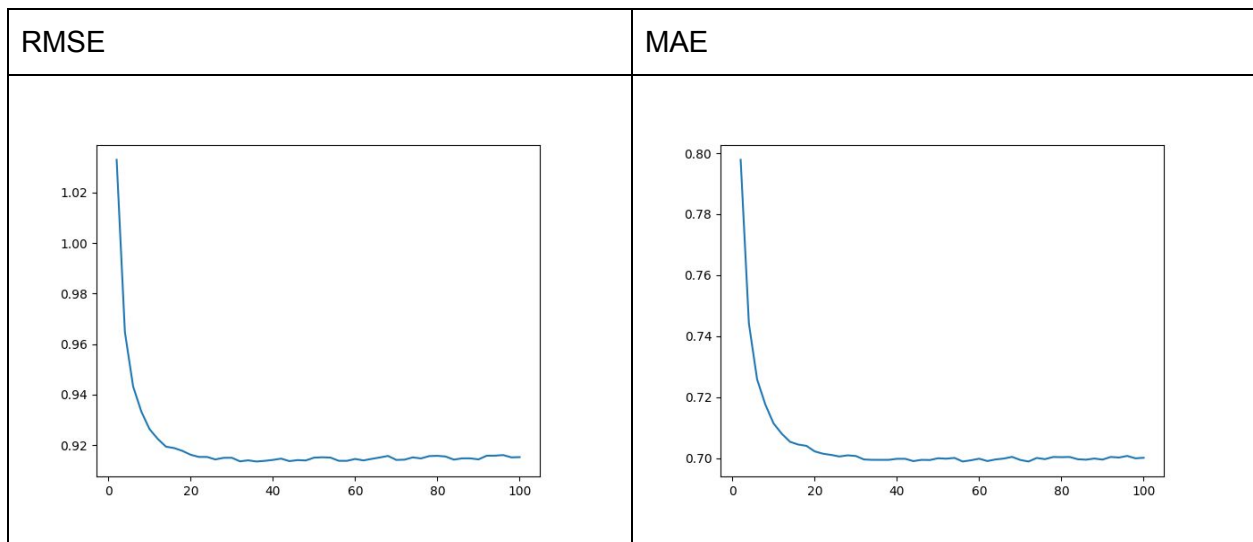
Q9

The use of  $\mu$  is to eliminate the bias of grading standard / threshold. To speak frankly, for example, person A and B may have the same taste among the movies. However, person A is a strict rater, while person B is a easy rater. In this circumstance, the two users would look very different without eliminate the bias. Therefore, we subtract the ratings by the mean rating to have a normalized data set.

## Q10

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

We design a k-NN collaborative filter to predict the ratings of the movies in the MovieLens dataset and evaluate its performance using 10-fold cross validation. The plots of RMSE and MAE measurement are shown as below.



The RMSE and MAE values are high in the beginning, but decrease sharply before reaching  $k=20$ . We found no significant difference in terms of RMSE and MAE from  $k=20$  to  $k=100$ .

## Q11

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

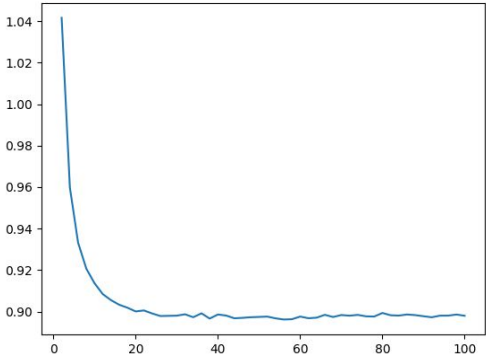
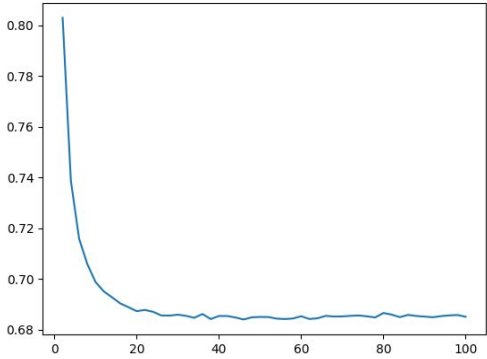
According to plots in Q10, the average of RMSE and the average of MAE both converge at around 20~30. Therefore, we analyze the difference between two continuous values and estimated the threshold as 0.0005. That is, when the decreasing in RMSE and MAE are less than 0.0005, we determine the k has converged. Finally, we got  $k=34$  for RMSE and  $k=36$  for MAE.

RMSE	MAE
0.916344347754	0.698536697655

## Q12

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

Use the popular trimmed set to estimate the RMSE and MAE value with kNN method.

RMSE	MAE
	
min: 0.8962	min: 0.6841

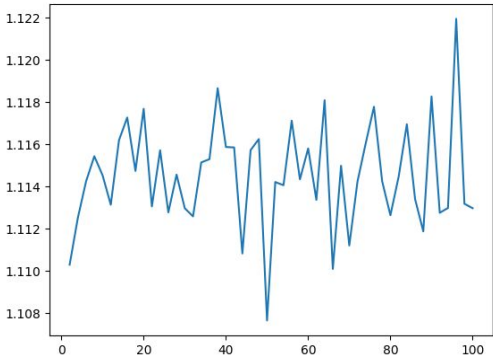
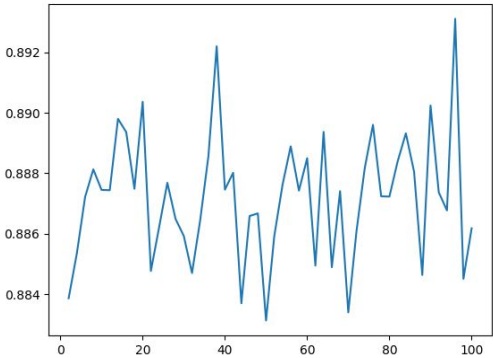
The graphs look similar to the graphs in Q10, except the fluctuation from k=20 to k=100 is slightly smoother.



## Q13

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

Use the unpopular trimmed set to estimate the RMSE and MAE value with kNN method.

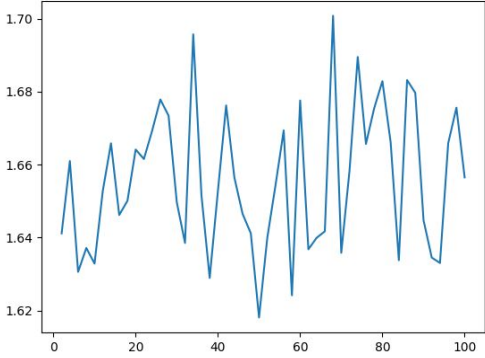
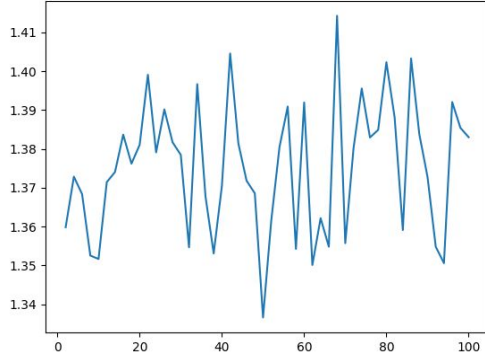
RMSE	MAE
	
min: 1.1077	min: 0.8311

The RMSE and MAE values are choppy regardless of the value of  $k$ , which is pretty reasonable since the unpopular dataset contains sparse data that is unpredictable.

## Q14

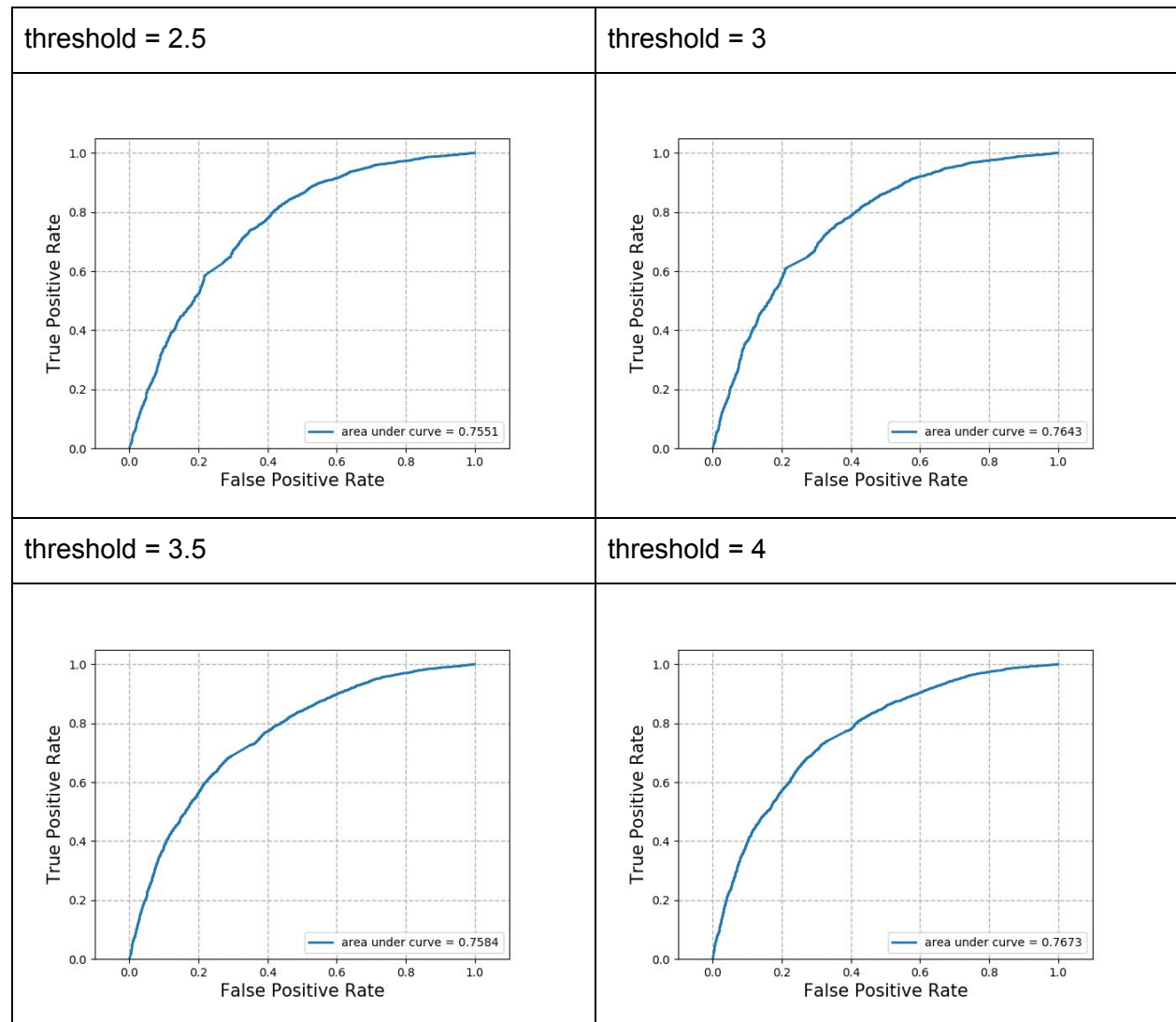
Question 14: Design a k-NN collaborative filter to predict the ratings of the movies in the high variance movie trimmed test set and evaluate its performance using 10-fold cross validation. Sweep k ( number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained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

Use the high-variance trimmed set to estimate the RMSE and MAE value with kNN method.

RMSE	MAE
	
min: 1.6181	1.3366

## Q15

Question 15: Plot the ROC curves for the k-NN collaborative filter designed in question 10 for threshold values [2:5; 3; 3:5; 4]. For the ROC plotting use the k found in question 11. For each of the plots, also report the area under the curve (AUC) value.



## Q30~Q33

In this section, Q30~Q33, we predict the rating of users among items via a primitive way, which is averaging. The rating will be just the mean rating of user  $i$ ,  $\mu_i$ . That means every new item respects to an user will be assigned the same value. Again, we splitted the data into 10 folds and check the accuracies by cross-validation.

In Q30, we will apply this on all movies, in Q31, we will apply this on only popular movies, and in Q32 we will apply this on unpopular movies. Furthermore, in Q33 ,we are interested in the movies with high variances of rating distribution, we will also perform this in the followings.

To perform the alogrithm, we define a class to achieve it.

```
from surprise import AlgoBase

class NaiveFiltering(AlgoBase):





    def __init__(self):

        # Always call base method before doing anything.
        AlgoBase.__init__(self)

    def estimate(self, u, i):

        if self.trainset.knows_user(u):
            return np.mean([r for (_, r) in self.trainset.ur[u]])
        else:
            return self.trainset.global_mean
```

Also, we have prepared the input data of Q30~Q32 as “.csv” files. The accuracies are shown below.

Q30	Q31	Q32	Q33
 ratings_new.csv	 ratings_popular.csv	 ratings_unpopular.csv	 ratings_var.csv
0.96245	0.95858	0.97896	0.95538

We can see that the results were good. That means, people tend to give similar scores among the movies, even the movies with high rating variances. The unpopular ones received the highest accuracy. This may be because of the number of the users were smaller.