Project 3: Recommender Systems

Large-Scale Data Mining: Models and Algorithms

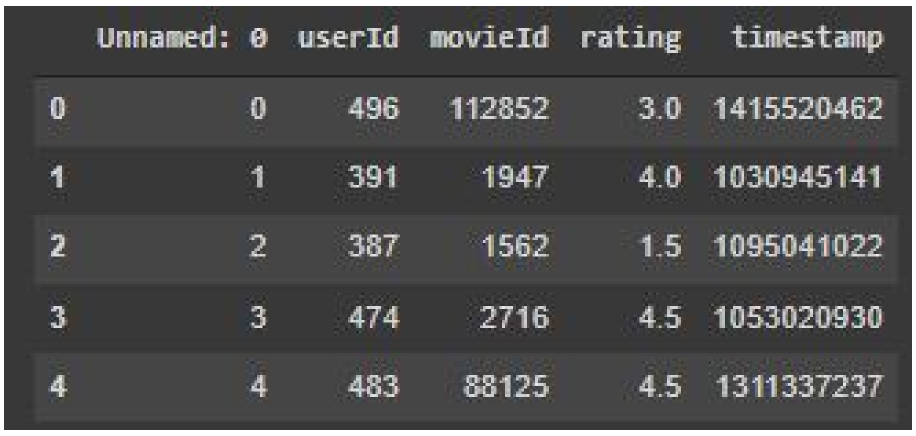
Ananya Deepak Deoghare, Hariram Veeramani, Madhav Sankar Krishnakumar

005627628, 005528336, 405692669

ECE 219 Winter 2022

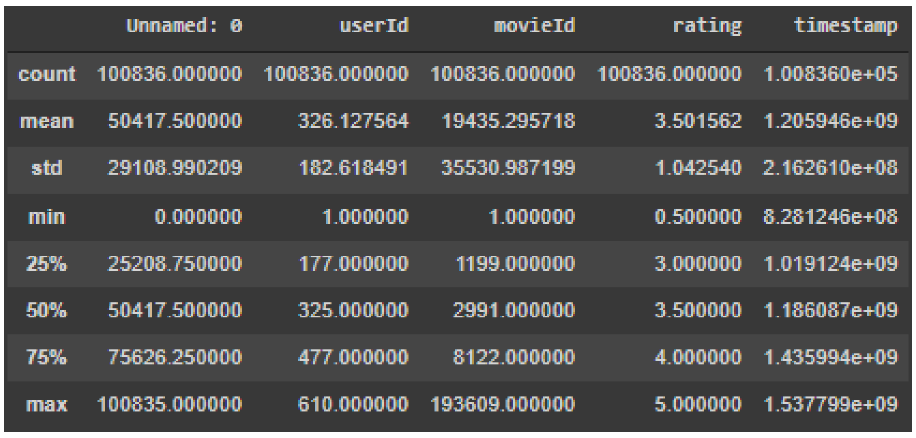
Question 1

A. Explore the Dataset: In this question, we explore the structure of the data. Compute the sparsity of the movie rating dataset:
Sparsity $= \frac{\textbf{Total number of available ratings}}{\textbf{Total number of possible ratings}}$
In this question, we explored the structure of the data for the Dataset MovieLens. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.
Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.



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

Figure 1: Output of Dataset.head()



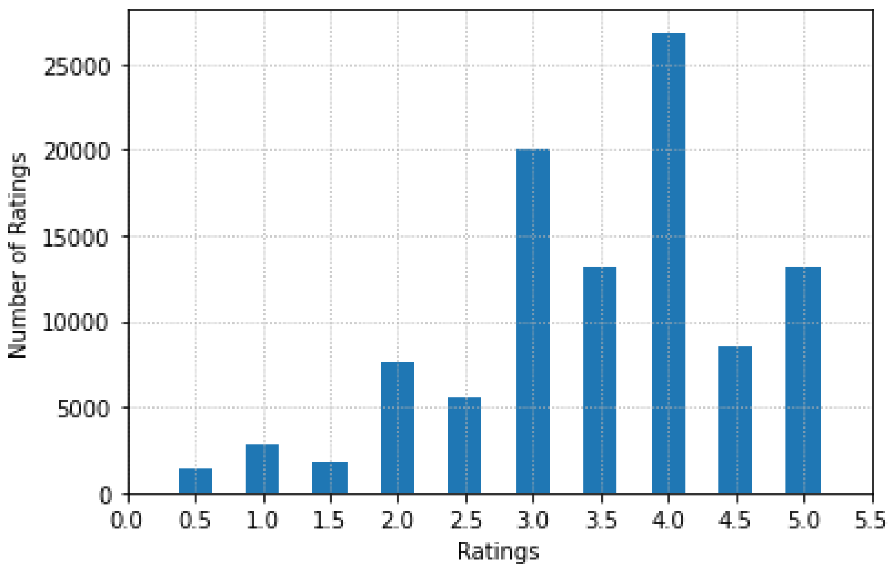
	Unnamed: 0	userId	movieId	rating	timestamp
count	100836.000000	100836.000000	100836.000000	100836.000000	1.008360e+05
mean	50417.500000	326.127564	19435.295718	3.501562	1.205946e+09
std	29108.990209	182.618491	35530.987199	1.042540	2.162610e+08
min	0.000000	1.000000	1.000000	0.500000	8.281246e+08
25%	25208.750000	177.000000	1199.000000	3.000000	1.019124e+09
50%	50417.500000	325.000000	2991.000000	3.500000	1.186087e+09
75%	75626.250000	477.000000	8122.000000	4.000000	1.435994e+09
max	100835.000000	610.000000	193609.000000	5.000000	1.537799e+09

Figure 2: Output of Dataset.describe()

The sparsity of a matrix is given using the below formula:
Sparsity $= \frac{\text{Total number of available ratings}}{\text{Total number of possible ratings}}$
It can also be defined as,
Sparsity $= \frac{\text{Total number of available ratings}}{\text{Total number of users}\times\text{Total number of movies}}$
The Sparsity of MovieLens dataset = 0.016999683055613623. The low value of sparsity indicates that the resultant matrix is sparse.
Each user is assigned a row in the resultant matrix and each movie will be assigned a column in the resultant matrix, therefore, each element denotes the rating for the movie given by the user.If the value of sparsity is very low as in general, it implies that the number of active users only rated a small portion of the movies.

B. Plot a histogram showing the frequency of the rating values: Bin the raw rating values into intervals of width 0.5 and use the binned rating values as the horizontal axis. Count the number of entries in the ratings matrix R that fall within each bin and use this count as the height of the vertical axis for that particular bin. Comment on the shape of the histogram.

To plot the histogram showing the frequency values, we binned the rating values into intervals of width 0.5 and used the binned values as the horizontal axis. We then counted the number of entries in the rating matrix R with ratings in the intervals and used this count as the vertical axis.



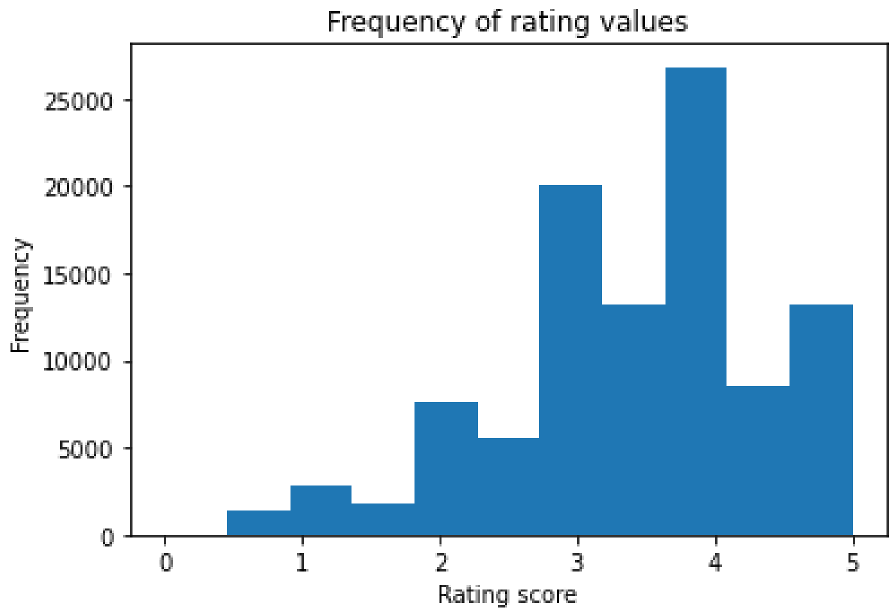


Figure 3: Frequency of rating values

We notice that the rating values in each interval is not distributed evenly. This is indicative of the fact that more movies were rated highly than lowly. We see that most users have rated movies which have rating values from 3.0 to 5.0, so most users liked the movies they watched. It is intuitively reasonable that popular movies or movies that get a lot of hype or movies with higher ratings from well-known websites and sources are often watched by a large number of users, and hence get rated more than the unpopular movies. As a result, the recommendation system tends to recommend popular movies to other users. Also, there is a user tendency to not rate the movies if they did not like it. This could also explain the shape of the histogram. In addition, integer ratings have higher counts than fractional ratings as most humans think of numbers as integers rather than as fractions.

C. Plot the distribution of the number of ratings received among movies: The X-axis should be the movie index ordered by decreasing frequency and the Y -axis should be the number of ratings the movie has received; ties can broken in any way. A monotonically decreasing trend is expected.

To plot the distribution of the number of ratings each movie received, we first ordered the movies by decreasing frequency, i.e., the movies with largest number of ratings has index 1 and this index is used as the X-axis. The Y-axis is simply the number of ratings for each movie. We obtained a monotonically decreasing curve as expected.

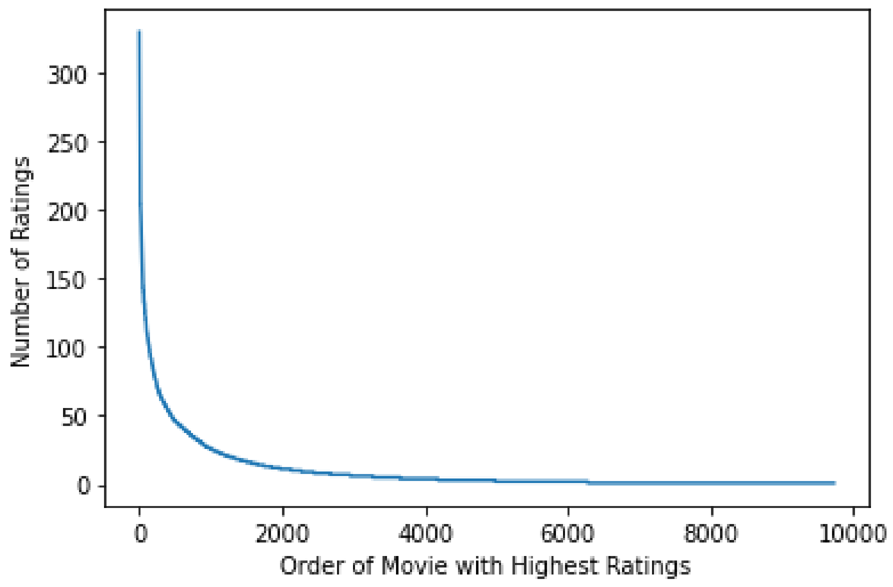


Figure 4: Distribution of the number of ratings received among movies

The largest number of ratings received per movie is 329 and the minimum is 1. Around 20% of the movies have received fewer than 10 ratings. Approximately 500 movies out of 9742 received more than 50 unique user ratings. This explains the sparsity of the ratings matrix, with only a few movies receiving multiple unique ratings. This may lead to a cold start problem due to which the recommendation system may fail.

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

The plot below explains the distribution of ratings among users, with the X-axis being user index ordered by decreasing frequency and the Y-axis is the number of movies the user has rated. We noticed a monotonically decreasing curve.

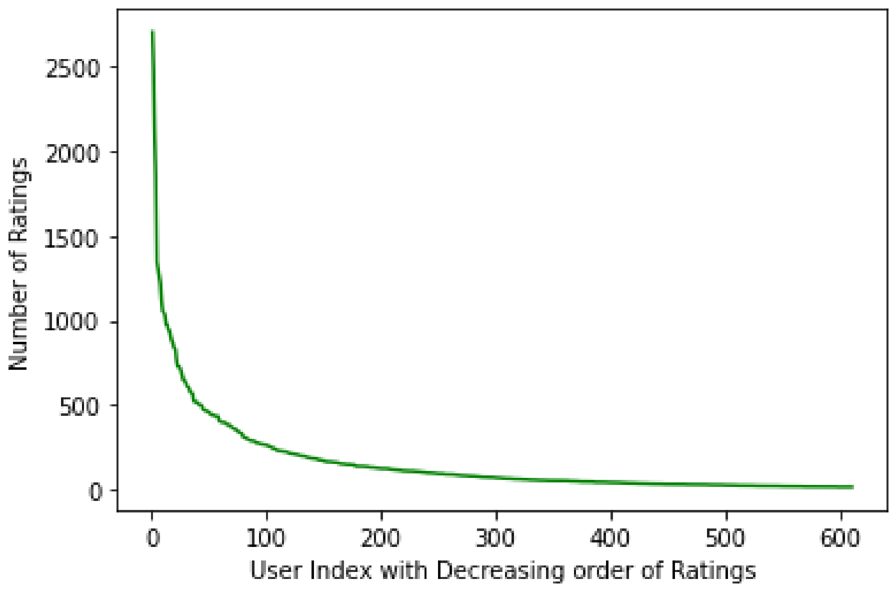


Figure 5: Distribution of ratings among users

The highest number of ratings given by one user is 2698. And similar to the previous subpart we notice that only around 20% users rate movies in this dataset. Less than 50 users out of 610 provide ratings to 500 movies or more out of 9742.

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

From parts C and D, we observe the monotonically decreasing curve.
Part C: The number of ratings has a reciprocal relationship with the movie index. This means that a small number of movies received a majority of the ratings. Only about 500 movies out of the 9742 received more than 50 user ratings. This implies that a lot of movies received a very small number of ratings. Around 20% of the movies received fewer than 10 ratings.
Part D: The number of ratings has a reciprocal relationship with the user index. This means that a small number of users accounted for most of the ratings. Highest number of rating by a user was a staggering 2698. This implies that a lot of users hardly rated. Less than 50 users out of 610 provided ratings for more than 500 movies.

Hence, this implies that the rating matrix R is sparse which means heavy regularization needs to be added to the recommendation process to prevent overfitting and false links.

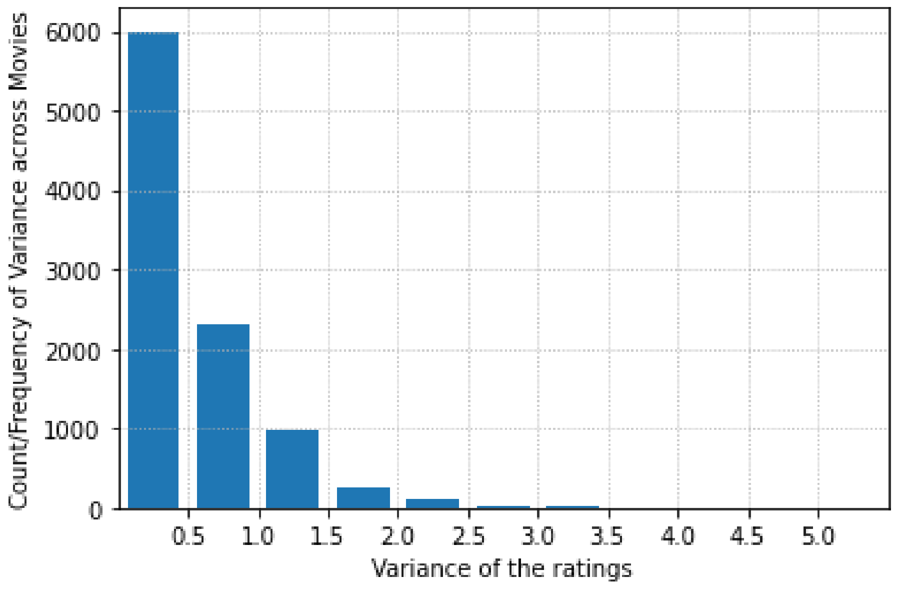
It is intuitively reasonable that popular movies or movies that get a lot of hype or movies with higher ratings from well-known websites and sources are often watched by a large number of users, and hence get rated more than the unpopular movies. This explains why most of the ratings are only for the few popular movies. As a result, the recommendation system tends to recommend popular movies to other users.

Along the similar lines, we can also explain the user scenario. A few users could be very active regarding watching movies and tend to watch more movies than others. In fact, only a part of these users might have the habit of rating the watched movies as well. This leads to a sparsity from the user aspect as well where there is a lot of data for some users and hardly any for others as observed in 1D.

From a ML perspective, the network or architecture will be biased to the popular/highly rated movies and towards users who have rated more. The Recommender systems face a problem in recommending items to users in case there is very little data available related to the user or movie. This is also known as the cold start problem. The low interaction between user and movie is one of the main reasons why the Recommender system can fail. If only few records are available, a collaborative algorithm can recommend it, but the quality of those records will be poor.

F. Compute the variance of the rating values received by each movie: Bin the variance values into intervals of width 0.5 and use the binned variance values as the horizontal axis. Count the number of movies with variance values in the binned intervals and use this count as the vertical axis. Briefly comment on the shape of the resulting histogram.

We computed the variance of the rating values received by each movie and binned the variance values into intervals of width 0.5 and used the binned variance values as the horizontal axis. The count of movies with variance in each binned intervals are used as the vertical axis. We used the standard np.var function with default ddof of 0.



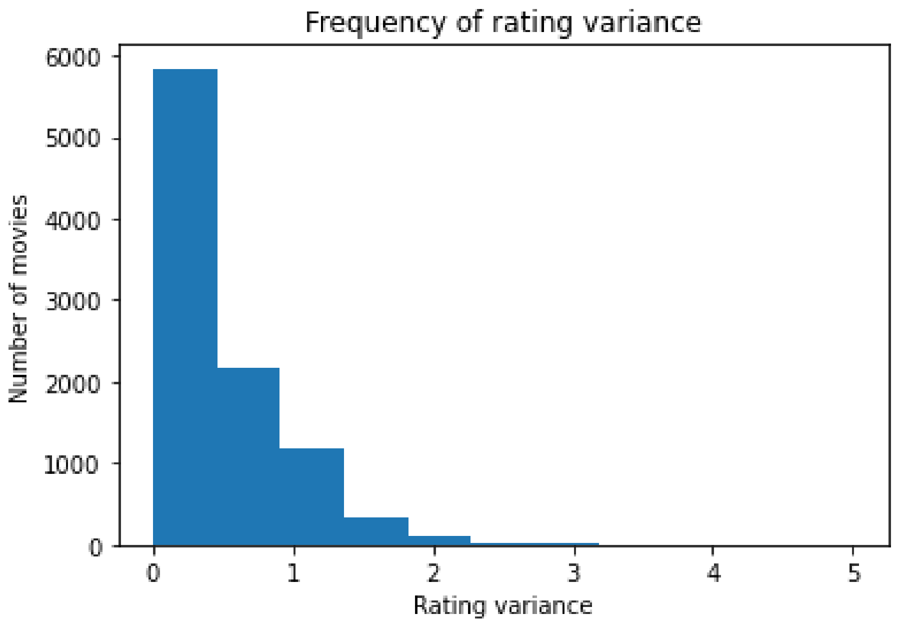


Figure 6: Variance of rating values received by each movie

We notice a decreasing trend in the variance. This shows that more than half of the movies received very similar ratings and almost no movies received ratings that differ more than 2.5. The decreasing tend makes sense as users trend to have similar judgements on movies. As most of the movie variance lie between 0 and 2, we can conclude that most ratings are reliable and consistent.

Question 2

Understanding the Pearson Correlation Coefficient:
A. Write down the formula for μ_u in terms of I_u and r_{uk};
The notations to be used are defined as below:
I_u: Set of movie indices for which ratings have been specified by user u
I_v: Set of movie indices for which ratings have been specified by user v
μ_u: Mean rating for user u computed using his/her specified ratings
r_{uk}: Rating of user u for movie k
From the definition provided to us in the question we wrote the formula for I_u and r_{uk}:

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

B. In plain words, explain the meaning of $I_u \bigcap I_v$. Can $I_u \bigcap I_v = \emptyset$? (Hint: Rating matrix R is sparse)

$I_u \bigcap I_v$ indicated the set of movies for which ratings have been specified by both user u and user v. $I_u \bigcap I_v$ can be null for the MovieLens dataset. From the previous section, we can know that around 500 movies received more than 50 unique user ratings. There is a very high probability that there may be many movies that were rated by user u but not by user v and vice versa. Therefore, the extreme end of 2 users not sharing any overlap of movie ratings is also possible.

As ratings matrix, R is sparse, it is possible that user u and v did not watch or rate any same movie. In which case $I_u \bigcap I_v = \emptyset$. From Question 1D, we can see that a large number of users rated only a handful of movies. So it is possible that these handful of movies have a 0 intersection set.

Question 3

Understanding the Prediction function: Can you explain the reason behind mean-centering the raw ratings $(rvj - \mu_v)$ in the prediction function? (Hint: Consider users who either rate all items highly or rate all items poorly and the impact of these users on the prediction function.)

Mean-centering will reduce the impact of different rating habits of users. For example, user u and user v have similar movie tastes, but one may be stricter in giving the ratings than the other. If we remove the mean-centering, there is a probability that the Recommender System may give inaccurate predictions.

Mean-centering helps us reduce user-specific bias and outliers which helps make the data less noisy. For example, users who either rate all items highly or poorly are usually giving extreme opinions which is biased and can be considered noisy. Thus, we can make a more accurate prediction if we mean-center the ratings. It helps us find the significance of the rating given by the user.

Question 4

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

We designed a k-NN collaborative filter (CF) to predict the ratings of the movies in the MovieLens dataset. We evaluated it's performance using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of neighbors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) and average MAE (Mean Absolute Error) across all 10 folds.

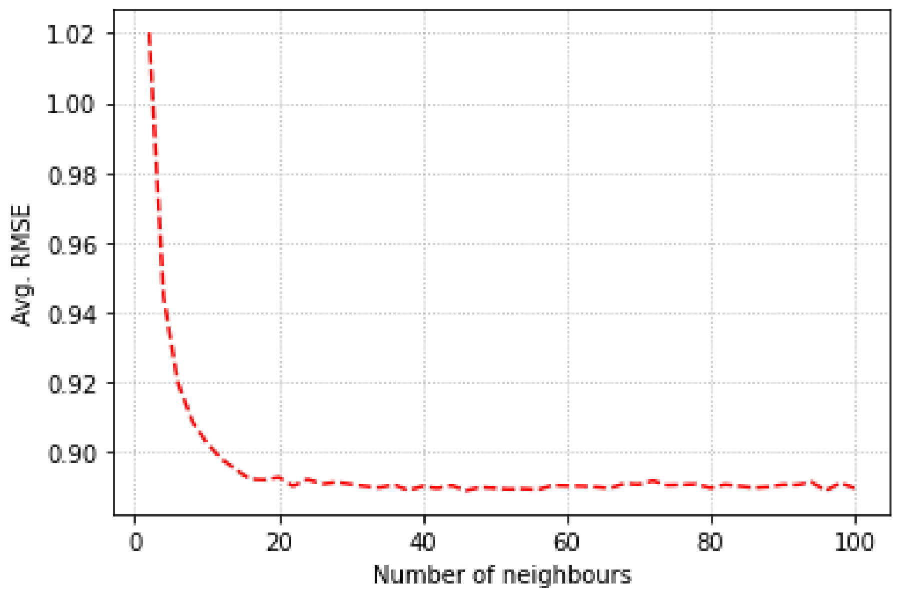


Figure 7: RMSE Vs Number of neighbors (k)

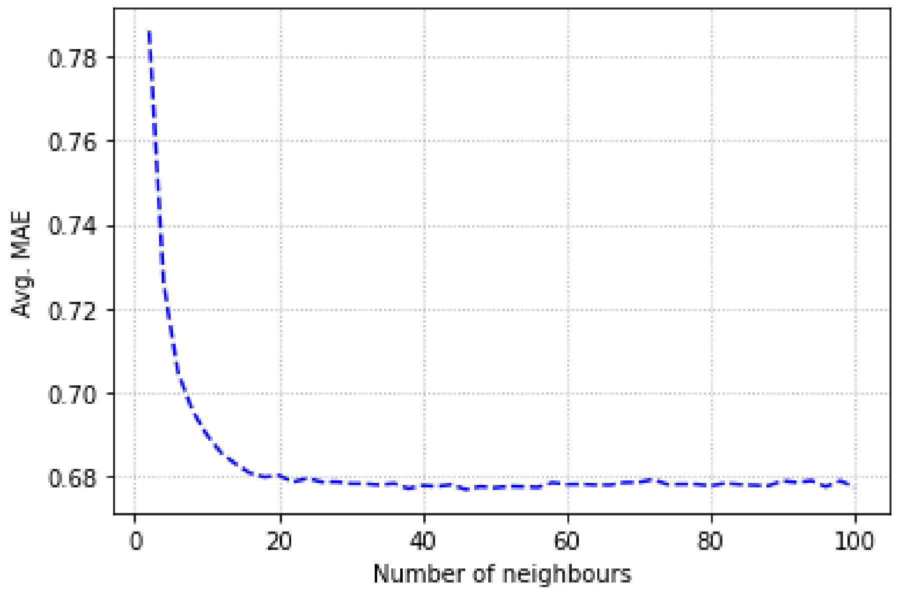


Figure 8: MAE Vs Number of neighbors (k)

We investigated the link between prediction error and the number of neighbors. By utilizing the correlation of ratings between users or objects from a sparse matrix of user ratings, Collaborative Filter analyzes historical ratings across users for similar items to anticipate items that a user would be interested in. There are two types of neighborhood-based CF model:

- User-based: CF that exploits similarity across users to predict ratings on new items.
- Item-based: CF based on the hypothesis that similar items are rated similarly by the same user.

User-based approach is often harder to scale because of the dynamic nature of users, whereas items usually don't change much, and item based approach often can be computed offline and served without constantly re-training. We are using user-based CF with K-NN. KNN is a non-parametric, lazy learning method. It uses a database in which the data points are separated into several clusters to make inference for new samples. KNN does not make any assumptions on the underlying data distribution but it relies on item feature similarity. When KNN makes inference about a movie, KNN will calculate the "distance" between the target movie and every other movie in its database, then it ranks its distances and returns the top K nearest neighbor movies as the most similar movie recommendations.

- We use the weighted average of the ratings of the k similar users to predict ratings for the target user on target item.
  The k-Nearest neighbors of user u, denoted by Pu, is the set of k users with the highest Pearson-correlation coefficient with user u (pairwise). The predicted rating that user u might award for item j, denoted by $\hat{r}_{uj}$, μ_x provides the mean rating for user x computed using all the ratings (useful to mean centering).

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

$$\mu_x = \frac{\sum_{k \in I_x} r_{xk}}{|I_x|}$$

- We compute the similarity between the items using Pearson similarity metric.

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

From figures 7 and 8, we notice that as we increase k i.e the number of neighbors, the prediction error drops. We observe a nearly monotonically decreasing trend for both RMSE and MAE vs number of neighbors, when we increase k from 2 to 100 with a step count of 2. This indicates that prediction improves with kNN CF as it has more users in it's vicinity to compare the ratings with. This makes the prediction less prone to noise and outliers and hence more dependable.

Question 5

Use the plot from question 4, to find a 'minimum k'. Note: The term 'minimum k' in this context means that increasing k above the minimum value would not result in a significant decrease in average RMSE or average MAE. If you get the plot correct, then 'minimum k' would correspond to the k value for which average RMSE and average MAE converges to a steady-state value. Please report the steady state values of average RMSE and average MAE.

The Minimum value of k for which increasing k above the minimum value would not result in a significant decrease in average RMSE or average MAE are given below.

- **RMSE:**
  Looking at Figure 7, we can say that
  Minimum k = 20, steady state RMSE value = 0.89055
- **MAE:**
  Looking at Figure 8, we can say that
  Minimum k = 20, steady state MAE value = 0.67892

We notice that increasing k beyond 20 does not really help much in decreasing the error and hence is just additional computation that can be avoided. This is because by k = 20, we have enough neighbors to generate an accurate prediction even with outliers and noise.

Question 6

A k-NN collaborative filter to predict the ratings of the movies in the test subset (i.e Popular, Unpopular or High-Variance) and evaluate each of the three models' performance using 10-fold cross validation.
Sweep k (number of neighbors) from 2 to 100 in step sizes of 2, and for each k compute the average RMSE obtained by averaging the RMSE across all 10 folds. Plot average RMSE (Y-axis) against k (X-axis). Also, report the minimum average RMSE.
We designed a k-NN collaborative filter to predict the rating of the movies in the below test subsets. The models' performances were evaluated using 10 fold cross validations. The step by step procedure for the same is given below:

1. For each value of k, split the dataset into 10 pairs of train and test sets.
   (train set 1, test set 1), (train set 2, test set 2), ... (train set 10, test set 10)
   We use the KFold() function in surprise scikit learn library to implement the 10 fold cross validation. Surprise provides various tools to run cross-validation procedures and search the best parameters for a prediction algorithm.
2. For each pair of train and test set:
   - Train the CF on the train set.
   - Call the required trimming filter as the input to the test set and the outputs the trimmed test set.
   - Predict the ratings of movies in the trimmed test set using the trained CF.
   - Compute the RMSE of the predictions in the trimmed set.
3. We test the CF with Pearson similarity metric for various number of neighbors, training on the training folds and testing on the trimmed set.
4. Compute the average RMSE by averaging across all 10 folds.

- **Popular movie trimming:**
  In this trimming, we trim the test set to contain movies that have received more than 2 ratings in the original dataset. If a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  We evaluated the performance of popular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of neighbors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

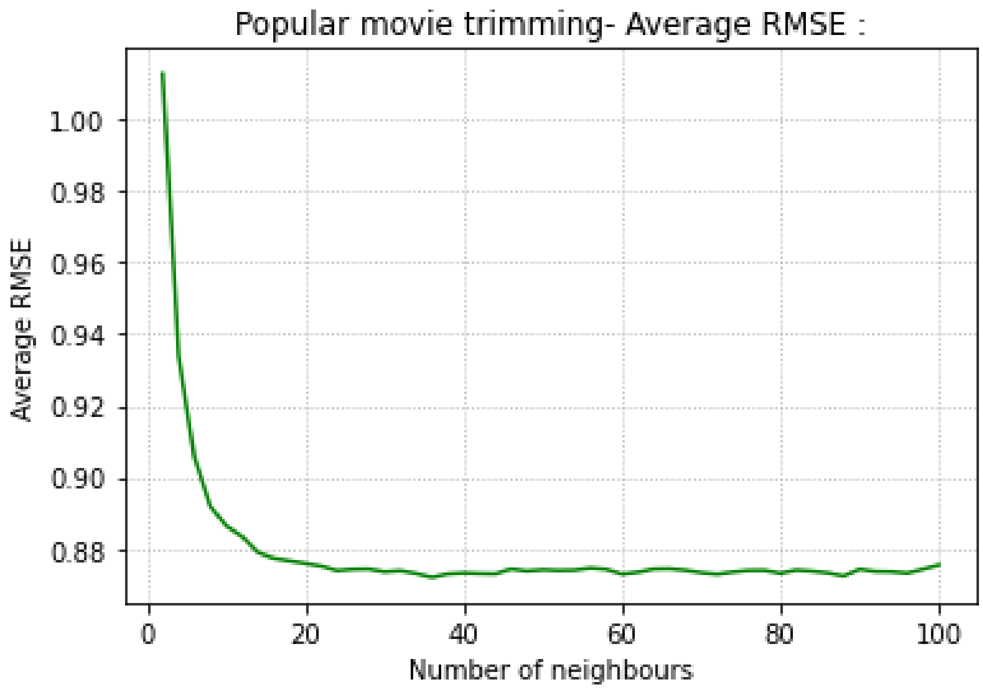


Figure 9: Average RMSE vs k for K-NN user-based CF on popular movie trimmed test set.

The minimum average RMSE for Popular movie trimming test set: **0.8719871055017627.** Absolute minimum is observed for k = 36. As it reached a steady state value at k = 20, we can take that as the minimum value going forward.
From the above figure, we notice a monotonically decreasing curve similar to the one we noticed in Figure 7. As expected we notice a much lower prediction error as compared to the values in Fig 7. This is due to the fact that we have trimmed out movies with fewer ratings, thus removing a few outliers.

- **Unpopular movie trimming:**
  In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings in the entire dataset. If a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  We evaluated the performance of unpopular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of neighbors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

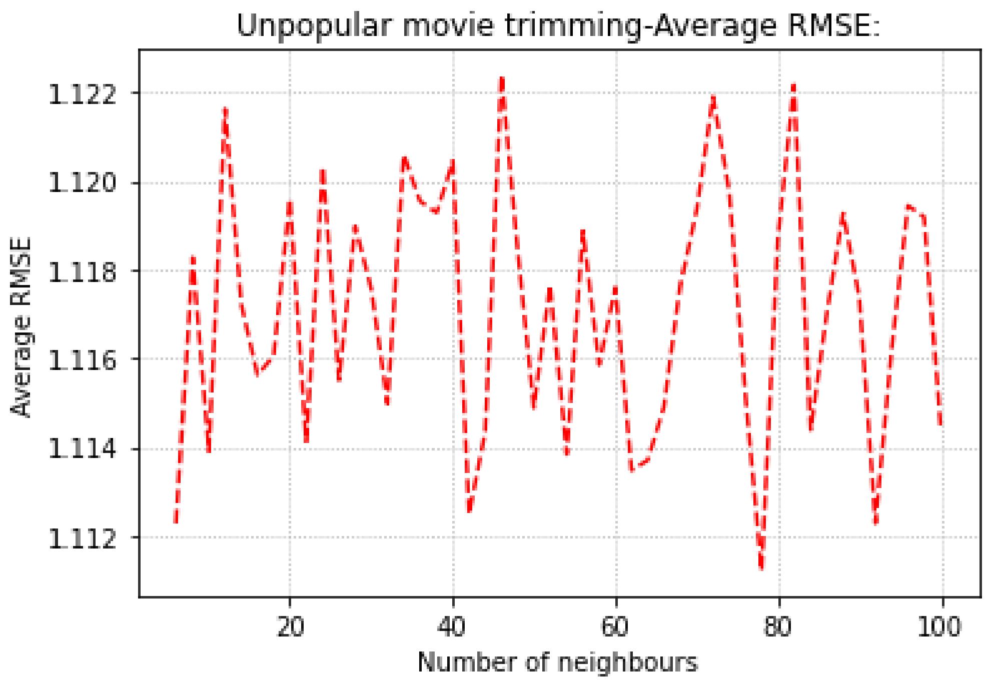


Figure 10: Average RMSE vs k for K-NN user-based CF on unpopular movie trimmed test set.

The minimum average RMSE for Unpopular movie trimming test set: **1.1112045741730798**. This is observed for k = 78. At k = 20, we observe **1.11882**.
We notice that the above RMSE value is higher than that we obtained for the popular movie trimmed set and the original test set. We can claim that the k-NN Collaborative Filter model is not viable for the unpopular movie trimmed test set, because all of the movies left in the test dataset after trimming earn low user ratings. As a result, locating neighbors in the training dataset is difficult, if not impossible, due to the sparseness of the relevant column of the rating matrix. Because the predictor was trained on the full dataset, which includes popular movies, it has trouble accurately forecasting the ratings of "unusual" things because there aren't enough individuals in the area who have rated them.
We notice a jumpy pattern in the above figure. Rather than expressing the link between RMSE and different selections of k, the jumpy pattern in the above figure is created by distinct data splitting schemes rendered by different random states in cross-validation. Furthermore, the difference between RMSE is so small that the relevant curve is really a horizontal line if we partition the entire dataset in the same way every time.

- **High variance movie trimming:**
  In this trimming, we trim the test set to contain movies that have variance (of the rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. If a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  We evaluated the performance of High variance movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of neighbors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

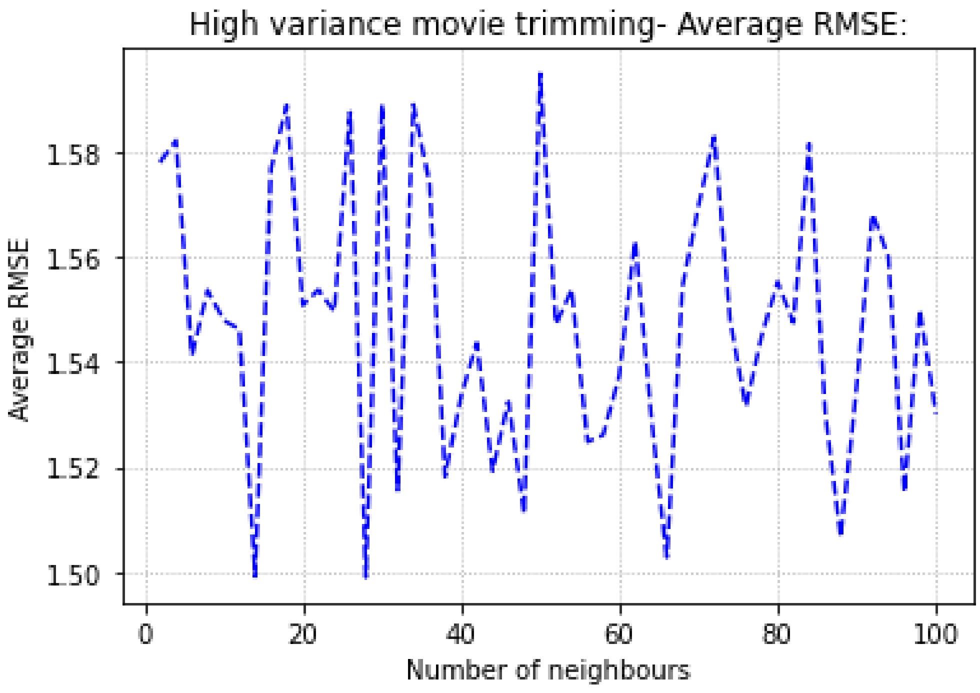


Figure 11: Average RMSE vs k for K-NN user-based CF on High variance movie trimmed test set.

The minimum average RMSE for High variance movie trimming test set: **1.499004534919353**. This is observed for k = 28. At k = 20, we observe **1.5583**.
The same scenario as in the shortened test set for an unpopular movie applies here. According to Question 1F, just a few films have a wide range of ratings. As a result, predicting on the test dataset after high variance movie trimming will result in the same issue as before: there will be insufficient neighbors in the training dataset, and the only option is to predict using the global mean.
This range of values seems to be a little higher than that of unpopular trimming.

Plot the ROC curves for the k-NN collaborative filters for threshold values [2.5, 3, 3.5, 4]. For each of the plots, also report the area under the curve (AUC) value.
We plot the ROC curve for the k-NN Collaborative filters for the threshold values [2.5, 3, 3.5, 4]. We split the data into train and test, with a 90% and 10% split respectively. We used the k value found in the previous question i.e., k = 20. We also assumed that if the user likes the movie (rating is equal to or higher than a threshold), the label = 1.
The above ROC curve shows the plot for true positive rate Vs false positive rate. It is a measure of the relevance of the film recommended to the user.

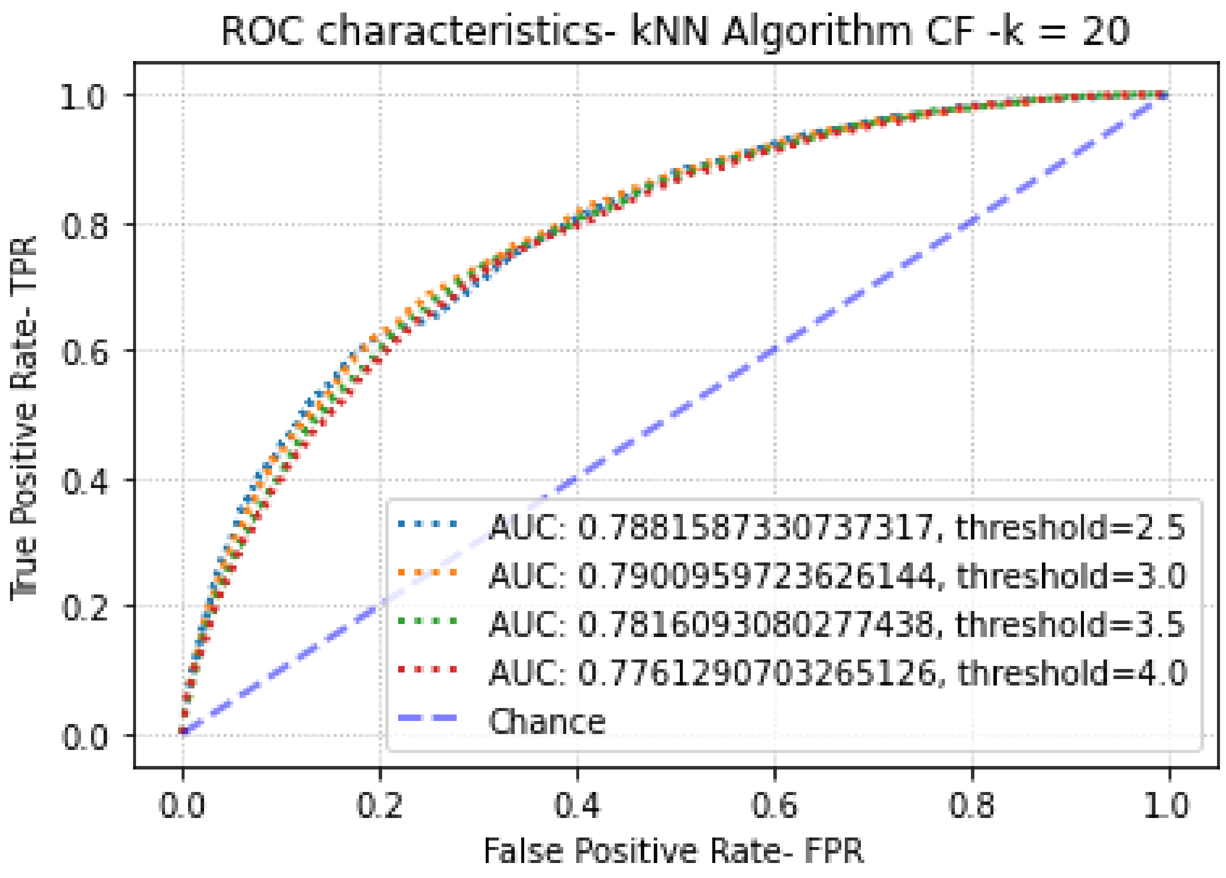


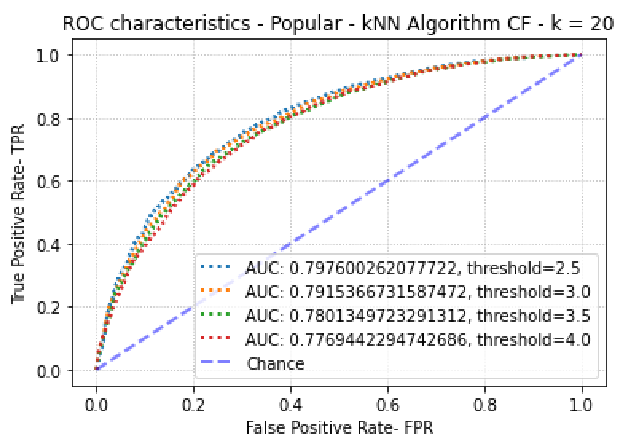
Figure 12: ROC curves for various thresholds for k-NN user-based CF with 20 neighbors

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values.

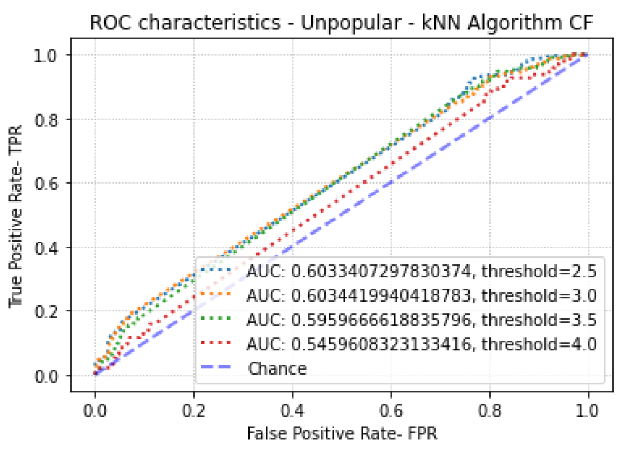
- **Threshold 2.5:** 0.78815
- **Threshold 3:** 0.79009
- **Threshold 3.5:** 0.78160
- **Threshold 4** 0.77612

Threshold 3 renders highest AUC scores, which is 0.79009.

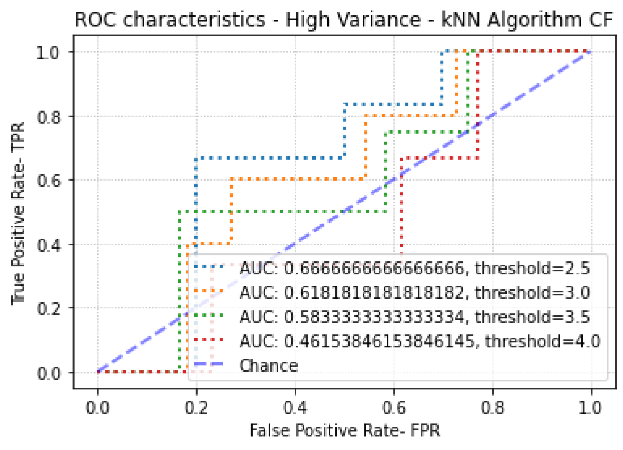
Now let us plot the ROC curves for various different trimmed subsets.



(a) ROC for Popular - kNN Algorithm CF.



(b) ROC for Unpopular - kNN Algorithm CF.



(c) ROC for High Variance - kNN Algorithm CF.

Figure 13: ROC curves for various trimmed subsets for k-NN user-based CF

We observe that the area under the ROC Curve for Popular case is better than the earlier one and this is understandable as well as the unpopular ones are removed and hence lesser outliers. It gives a best case AUC of 0.7976002 for threshold of 2.5. The unpopular and high variance scenarios have very poor AUC as expected in the range of 0.6. In case of unpopular the best AUC is 0.60344 (threshold = 3) and high variance case has AUC of 0.6666 (threshold = 2.5). We observe a step shaped ROC curve for high variance because of the lack of data points. As the data points are very low, a small change in the threshold beings about a huge change in the AUC as well.

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values for **Popular** trimming test set.

- **Threshold 2.5:** 0.79760
- **Threshold 3:** 0.79153
- **Threshold 3.5:** 0.78013
- **Threshold 4** 0.77694

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values for **Unpopular** trimming test set.

- **Threshold 2.5:** 0.60334

- **Threshold 3:** 0.60344
- **Threshold 3.5:** 0.59596
- **Threshold 4** 0.54596

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values for the **High Variance** trimming test set.

- **Threshold 2.5:** 0.66666
- **Threshold 3:** 0.61818
- **Threshold 3.5:** 0.58333
- **Threshold 4** 0.46153

Question 7

Understanding the NMF cost function: Is the optimization problem given by equation 5 convex? Consider the optimization problem given by equation 5. For U fixed, formulate it as a least-squares problem.

The NMF Cost function is given as:

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

To find if a function is convex or concave, we find the hessian matrix, i.e., double differentiate the function. If the Hessian matrix is not positive semi-definite, then the function is non-convex.

For simplicity, let's assume m = n = 1, $W_{11} = 1$. The above function reduces to:

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

Note: It doesn't make a difference if we multiply the matrix with some constant. Calculating the Hessian Matrix of L wrt U,V:

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

The determinant of the Hessian matrix is:

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

The above determinant is definitely not positive semi-definite. Therefore, for any general case of m,n the objective function is non-convex.

Also, there are many local minimas in the objective function gradient plane. Therefore, the optimization problem is not simultaneously convex for user latent space, U, and item embedding space, V. This is due to the fact that the matrix factorization model predicts ratings using the combination of U and V, which does not satisfy the convexity property because the objective function is permutation and rotation invariant.

The optimization problem for least-squares problem can be solved by keeping U fixed and solving for V or vice versa. For fixed U, (without regularization) the objective function becomes,

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

$V = (UU^T)^{-1}UR$

where R = ratings matrix

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

Question 8

Designing the NMF Collaborative Filter

A. Design a NMF-based collaborative filter to predict the ratings of the movies in the original dataset and evaluate it's performance using 10-fold cross-validation. Sweep k (number of latent factors) from 2 to 50 in step sizes of 2, and for each k compute the average RMSE and average MAE obtained by averaging the RMSE and MAE across all 10 folds. If NMF takes too long, you can increase the step size. Increasing it too much will result in poorer granularity in your results. Plot the average RMSE (Y-axis) against k (X-axis) and the average MAE (Y-axis) against k (X-axis). For solving this question, use the default value for the regularization parameter.

We designed a NMF-based collaborative filter (CF) to predict the ratings of the movies in the MovieLens dataset. We evaluated it's performance using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) and average MAE (Mean Absolute Error) across all 10 folds. To implement this NMF-based collaborative filter, we utilize matrix factorization.NMF from surprise.prediction algorithms package.

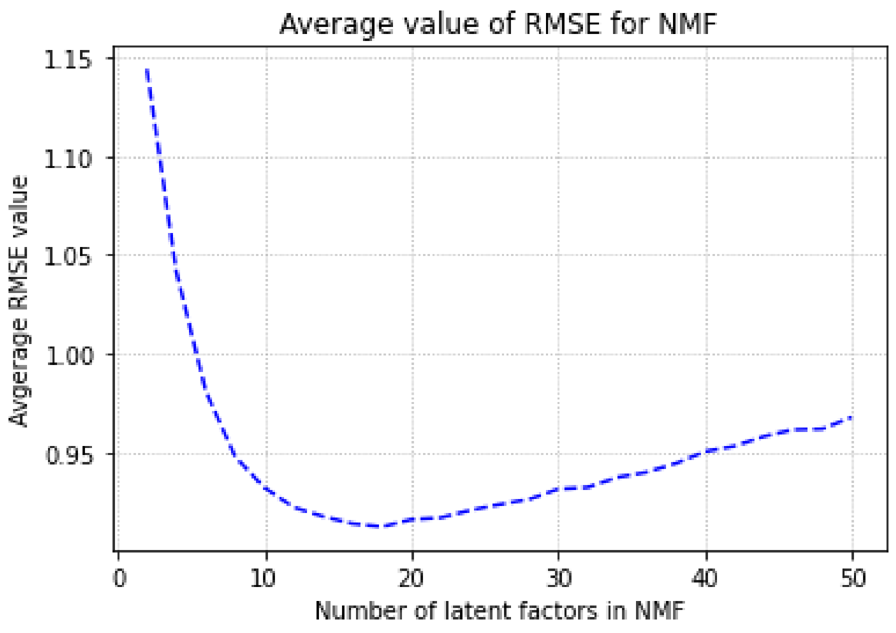


Figure 14: RMSE Vs Number of latent factors (k)

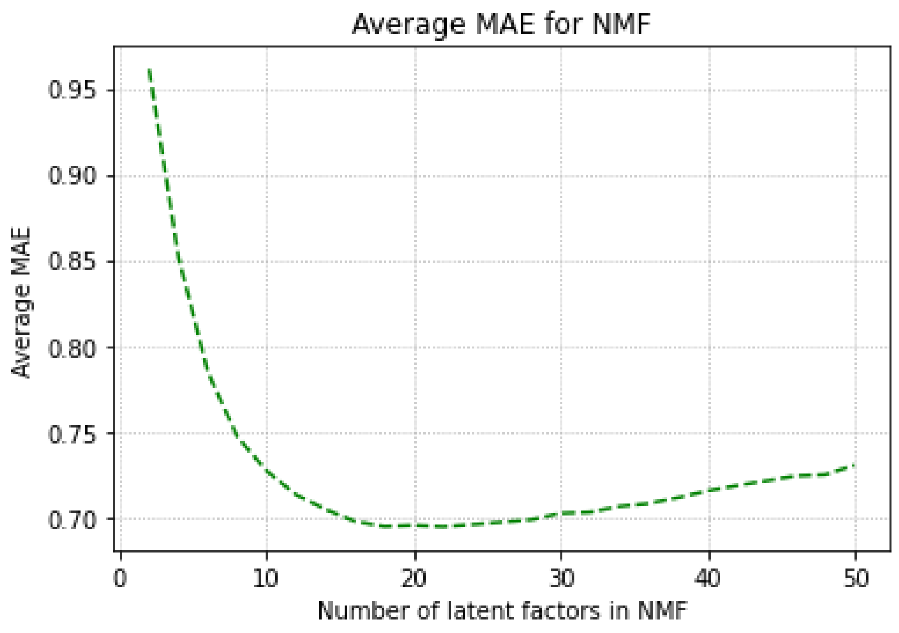


Figure 15: MAE Vs Number of latent factors (k)

Unlike k-NN, we don't observe a monotonically decreasing curve. We observe the values decrease with the minimum at around k = 20, and then the values start increasing.

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

Unlike k-NN, we don't observe a monotonically decreasing curve. We observe the values decrease with the minimum at around k = 20, and then the values start increasing. The Minimum value of k is given by:

- **RMSE:**
  Looking at Figure 14, we can say that
  Minimum k = 18, steady state RMSE value = 0.912530
- **MAE:**
  Looking at Figure 15, we can say that
  Minimum k = 22, steady state MAE value = 0.694350

There are 19 movie genres in the MovieLens Dataset, which is very close to the k values obtained by us. The average k value = (22+18)/2 = 20. In NMF, we try to find latent factors that can be generated from the users and that maps to the final movies. These latent factors tend to fit towards one or a few genres of the movies. Therefore, it makes sense that the number of latent factors tend to be close to the number of genres.

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

In this part, we desinged an NMF collaborative filter to predict the ratings of the movies in the trimmed test subset. We evaluated it's performance using 10-fold cross validation using KFold() function from the Surprise Skicit library. We swept k values from 2 to 100, with step size of 2. Average RMSE and Average MAE values were obtained across all 10 folds.

- **Popular movie trimming:**

  In this trimming, we trim the test set to contain movies that have received more than 2 ratings. If a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

  We evaluated the performance of popular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

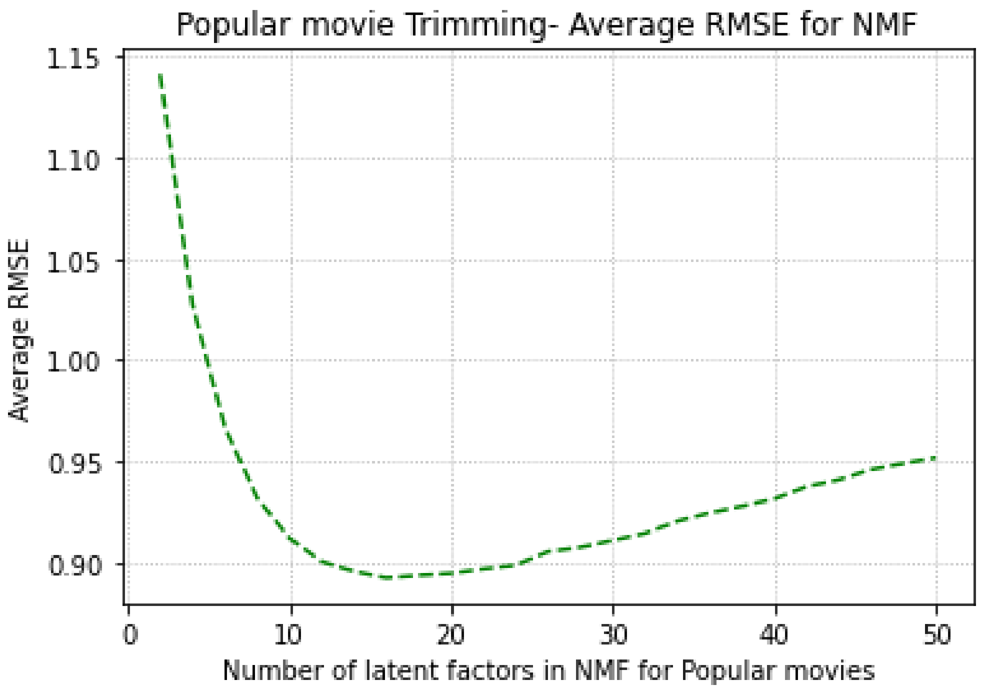


Figure 16: Average RMSE vs k for NMF user-based CF on popular movie trimmed test set.

The minimum average RMSE for Popular movie trimming test set: **0.8925320415716778**. From the above figure, we notice a curve similar to the one we noticed in Figure 14. As expected we notice a much lower prediction error as compared to the values in Fig 14. This is due to the fact that we have trimmed out movies with fewer ratings, thus removing a few outliers.

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

- **Unpopular movie trimming:**

  In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings. If a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

  We evaluated the performance of unpopular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

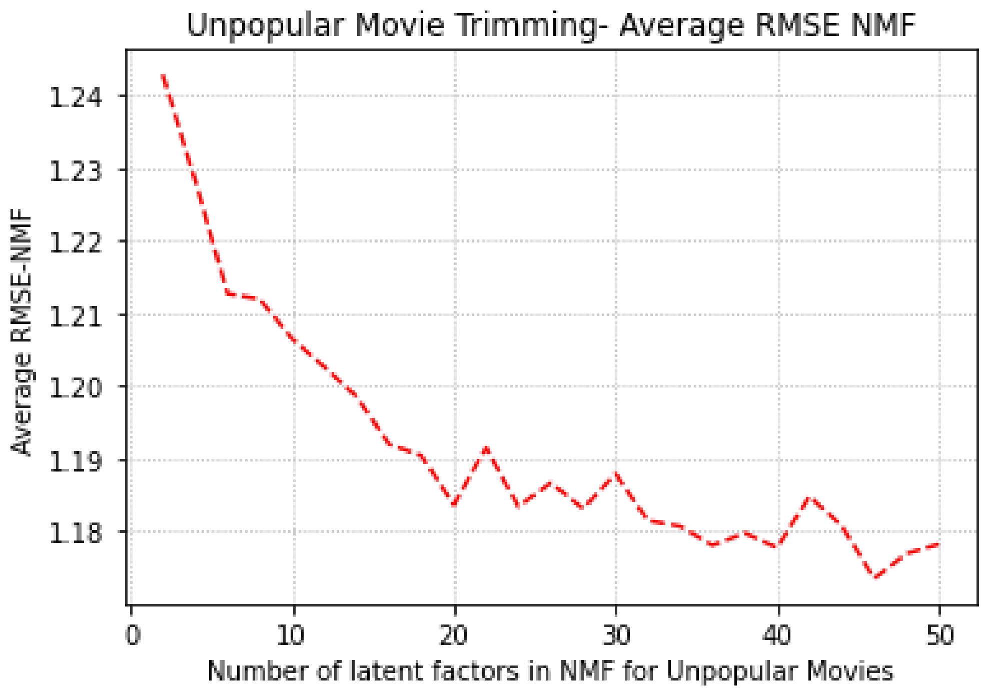


Figure 17: Average RMSE vs k for NMF user-based CF on unpopular movie trimmed test set.

  The minimum average RMSE for Unpopular movie trimming test set: **1.173498893400049**. At k = 18, we observe **1.1191**.

  We notice that the above RMSE value is higher than that we obtained for the popular movie trimmed set and the original test set. We can claim that the NMF Collaborative Filter model is not viable for the unpopular movie trimmed test set, because all of the movies left in the test dataset after trimming earn low user ratings. Because the predictor was trained on the full dataset, which includes popular movies, it has trouble accurately forecasting the ratings of ”unusual” things because there aren't enough individuals who have rated them. We notice a decreasing pattern instead of the jumpy pattern observed previously.

- **High variance movie trimming:**

  In this trimming, we trim the test set to contain movies that have variance (of the rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. If a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

  We evaluated the performance of High variance movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE

(Root Mean Square Error) across all 10 folds.

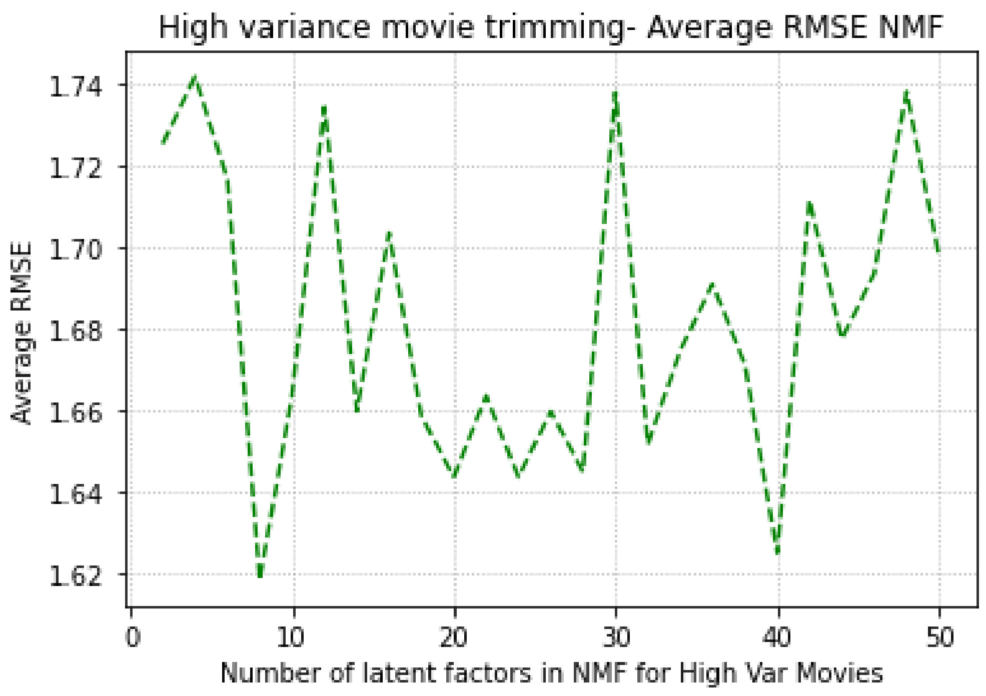


Figure 18: Average RMSE vs k for NMF user-based CF on High variance movie trimmed test set.

The minimum average RMSE for High variance movie trimming test set: **1.6182565408745588**.
At k = 18, we observe **1.6591**.
The same scenario as in the shortened test set for an unpopular movie applies here. According to Question 1F, just a few films have a wide range of ratings. As a result, predicting on the test dataset after high variance movie trimming will result in the same issue as before: there will be insufficient high variance data points in the training dataset. So it is hard for the model to provide accurate results on such outlier points.

The sorts of movies remaining in the testing dataset after high variance movie trimming are mostly contentious. Being controversial here implies that the film is somewhat popular (having a big number of ratings), yet the ratings supplied by various people disagree significantly (having large variance). As a result, predicting those films is difficult. That's why the plot has such erratic patterns.

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

We plot the ROC curve for the NMF Collaborative filters for the threshold values [2.5, 3, 3.5, 4]. We split the data into train and test, with a 90% and 10% split respectively. We used the k value found in the previous question i.e., k = 18. We also assumed that if the user likes the movie (rating is equal to or higher than a threshold), the label = 1.

The above ROC curve shows the plot for true positive rate Vs false positive rate. It is a measure of the relevance of the film recommended to the user.

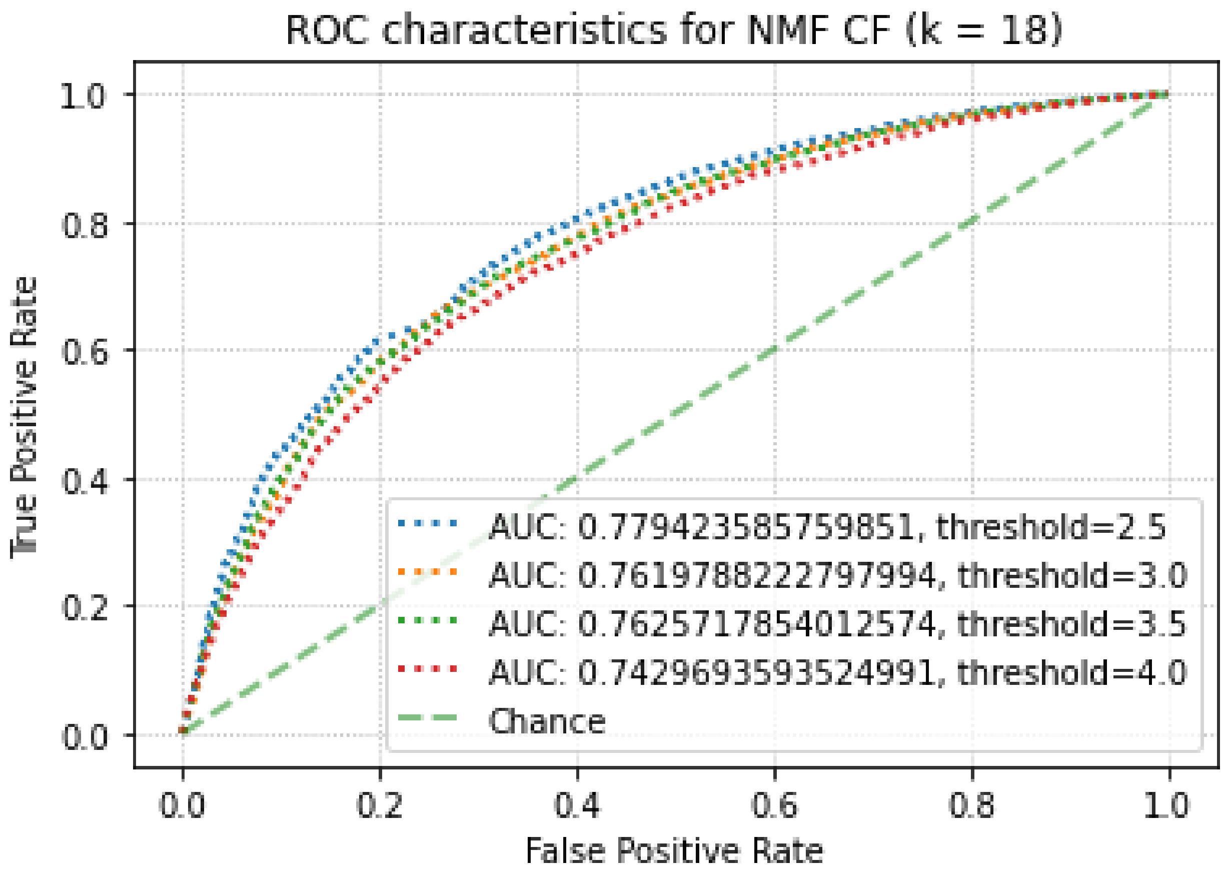


Figure 19: ROC curves for various thresholds for NMF user-based CF with k = 18 neighbors

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values.

- **Threshold 2.5:** 0.77942
- **Threshold 3:** 0.76197
- **Threshold 3.5:** 0.76257
- **Threshold 4** 0.74296

We notice that as we increase the value of threshold, the AUC decreases for NMF CF. Threshold 2.5 renders highest AUC scores, which is 0.77942.

Question 9

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

We performed Non-negative matrix factorization on the rating matrix R to obtain U and V. U represents the user-latent factors interaction and V represents the movie-latent factors interaction. We obtain the V matrix using NMF_K20.qi command where, NMF_K20 = NMF(n_factors=20,n_epochs=50,verbose=False)

We sorted out the rows of V in descending order and printed out the top 10 movies for random columns of V. The results for the same are:

```
Column number corresponding to V Matrix Instance:  1
Adventure
Horror|Mystery|Thriller
Children|Comedy
Documentary
Crime|Drama|Mystery|Thriller
Drama
Action|Adventure|Sci-Fi|IMAX
Comedy
Drama|Western
Comedy|Sci-Fi
Genres:  {'Documentary', 'Sci-Fi', 'Adventure', 'Comedy', 'Crime',
'Action', 'Children', 'Drama', 'Thriller', 'Horror', 'Western', 'Mystery', 'IMAX'}
Number of Unique Genres: 13
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  3
Action|Children|Sci-Fi|IMAX
Action|Drama|Sci-Fi
Horror
Action|Drama|War
Children
Drama
Horror|Mystery|Thriller
Comedy
Mystery|Thriller
Comedy|Drama
Genres:  {'Sci-Fi', 'Comedy', 'War', 'Action', 'Children',
'Drama', 'Thriller', 'Horror', 'Mystery', 'IMAX'}
Number of Unique Genres: 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  5
Drama|Horror
```

```
Action|Crime|Drama
Comedy|Crime
Adventure|Drama|Romance
Adventure|Drama|Thriller
Action|Comedy
Comedy
Comedy|Musical|Romance
Horror
Drama|Romance
Genres:  {'Adventure', 'Comedy', 'Crime', 'Action', 'Drama',
'Musical', 'Romance', 'Thriller', 'Horror'}
Number of Unique Genres: 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  7
Comedy
Comedy|Drama
Sci-Fi
Comedy
Crime|Drama
Documentary
Thriller
Comedy|Fantasy
Comedy
Adventure|Comedy|Drama|Fantasy|Romance
Genres:  {'Documentary', 'Sci-Fi', 'Adventure', 'Comedy',
'Crime', 'Drama', 'Romance', 'Thriller', 'Fantasy'}
Number of Unique Genres: 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  11
Action|Horror|Thriller
Documentary
Adventure|Drama|Western
Action|Crime|Thriller
Comedy
Drama|Western
Drama
Action|Drama|Horror
Action|Comedy
Adventure|Comedy|Crime
Genres:  {'Documentary', 'Adventure', 'Comedy', 'Crime',
'Action', 'Drama', 'Thriller', 'Horror', 'Western'}
Number of Unique Genres: 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  15
Action
Drama
Children|Fantasy
Drama|Fantasy|Romance
```

```
Adventure|Comedy|Fantasy
Comedy
Action|Adventure|Drama|Western
Adventure|Animation|Comedy
Comedy|Romance
Comedy|Drama
Genres:  {'Adventure', 'Comedy', 'Action', 'Children',
'Drama', 'Romance', 'Western', 'Animation', 'Fantasy'}
Number of Unique Genres: 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Column number corresponding to V Matrix Instance:  19
Crime|Drama|Thriller
Adventure|Drama|Romance
Horror
Comedy|Musical|Romance
Comedy
Drama
Comedy|Horror|Romance
Drama|Horror|Sci-Fi|Thriller
Western
Crime|Thriller
Genres:  {'Sci-Fi', 'Adventure', 'Comedy', 'Crime',
'Drama', 'Musical', 'Romance', 'Thriller', 'Horror', 'Western'}
Number of Unique Genres: 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

This shows the genres of the top 10 movies for particular latent factors. For latent factor 1, most of the top 10 movies are spread across dramas, crimes and action, indicating latent factor 1 represents a combination of a few genres. Similarly, latent factor 11 captures comedy and drama. While latent factor 15 is strongly tied to comedy. For all the latent factors, we observe that the top 10 movies belong to a small collection of genres. As the column number of the latent factor increases, the number of distinct movie genres decreases or remains the same. For example, when the number of latent factors = 1, the number of distinct movie genres is 13 and this comes down to 9 for 15 latent factors. Thus, movie genres for this instance seems to have clustered more closely together for the higher columns. When we re-ran the code multiple times, this observation was not always true.

The clustering property of NMF is evident. Each latent factor seems to cover a particular genre type or a small collection of the same. Hence, NMF clusters the movies amongst the various latent factor and this clustering seems to be largely based on genre.

Question 10

Designing the MF Collaborative Filter:

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

We designed a MF-based collaborative filter (CF) to predict the ratings of the movies in the MovieLens dataset. We evaluated it's performance using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of latent factors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) and average MAE (Mean Absolute Error) across all 10 folds. To implement this MF-based collaborative filter, we utilize matrix factorization.SVD from surprise.prediction_algorithms package.

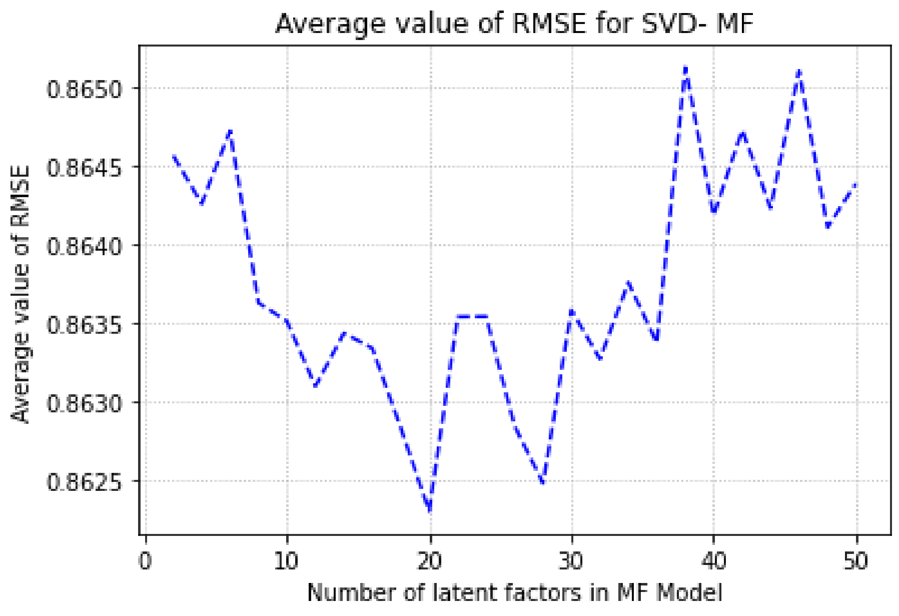


Figure 20: RMSE Vs Number of latent factors (k)

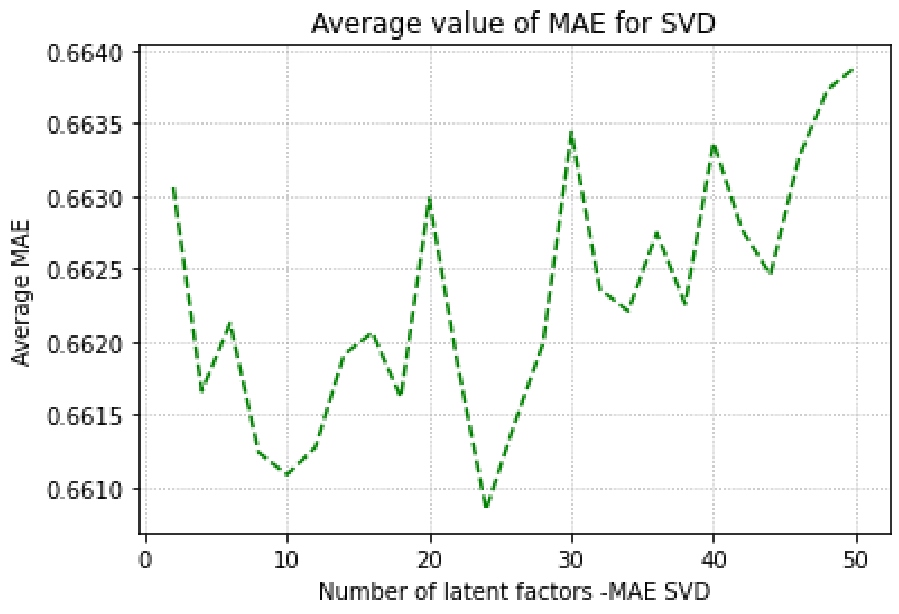


Figure 21: MAE Vs Number of latent factors (k)

The user bias term represents how a user's qualities cause them to offer polar evaluations when compared to the average, whereas the item bias term models how a specific object is rated when compared to the average ratings. Integrating bias information allows the produced embeddings to better model such user- or movie-specific noise or outliers. The optimization program is a singular value decomposition (SVD) issue.

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

The value of k for minimum average RMSE and average MAE are given below.

- **RMSE:**
  Looking at Figure 20, we can say that
  Minimum k = 20, steady state RMSE value = 0.862305
- **MAE:**
  Looking at Figure 21, we can say that
  Minimum k = 24, steady state MAE value = 0.660847

The ideal number of latent components based on both the average RMSE and the average MAE criterion is near the number of movie genres for MF with bias collaborative filter but this was not always the case. As a result, the latent components in this model are difficult to interpret. Furthermore, model results across a wide range of latent component counts are substantially equal. As a result, if we re-run cross-validation with a new random seed, the corresponding optimal number of latent factors is quite likely to be different. We assume that the value of k = 20. We notice that the optimal number in this case is quite close to the movie genre number i.e. 19.

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

In this part, we designed an MF-based collaborative filter to predict the ratings of the movies in the trimmed test subset. We evaluated it's performance using 10-fold cross validation using KFold() function from the Surprise Skicit library. We swept k values from 2 to 100, with step size of 2. Average RMSE and Average MAE values were obtained across all 10 folds.

- **Popular movie trimming:**

  In this trimming, we trim the test set to contain movies that have received more than 2 ratings. If a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.

  We evaluated the performance of popular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of latent factors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

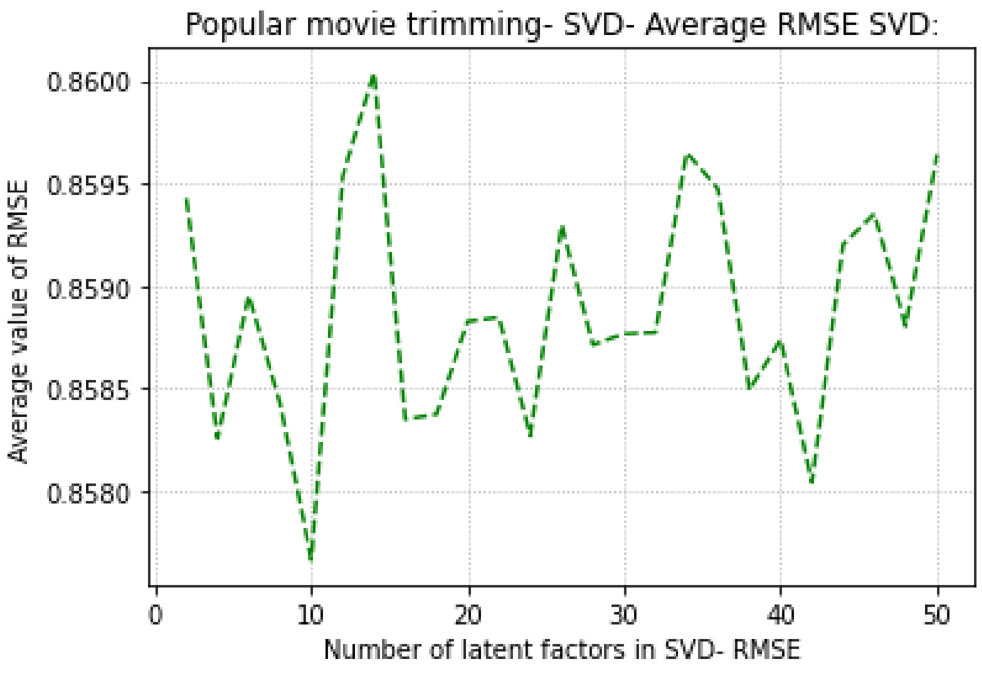


Figure 22: Average RMSE vs k for MF with bias (SVD) user-based CF on popular movie trimmed test set.

The minimum average RMSE for Popular movie trimming test set: **0.8576636998948253**. At k = 20, we observe **0.8588**.

From the above figure, we do not notice a monotonically decreasing curve like the one we noticed in earlier models. As expected we notice a much lower prediction error as compared to the values in Fig 20. This is due to the fact that we have trimmed out movies with fewer ratings, thus removing a few outliers.

We notice a jumpy pattern in the above figure. Rather than expressing the link between RMSE and different selections of k, the jumpy pattern in the above figure is created by distinct data splitting schemes rendered by different random states in cross-validation. Furthermore, the difference between RMSE is so small that the relevant curve is really a horizontal line if we partition the entire dataset in the same way every time.

- **Unpopular movie trimming:**
  In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings. If a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  We evaluated the performance of unpopular movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of latent factors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

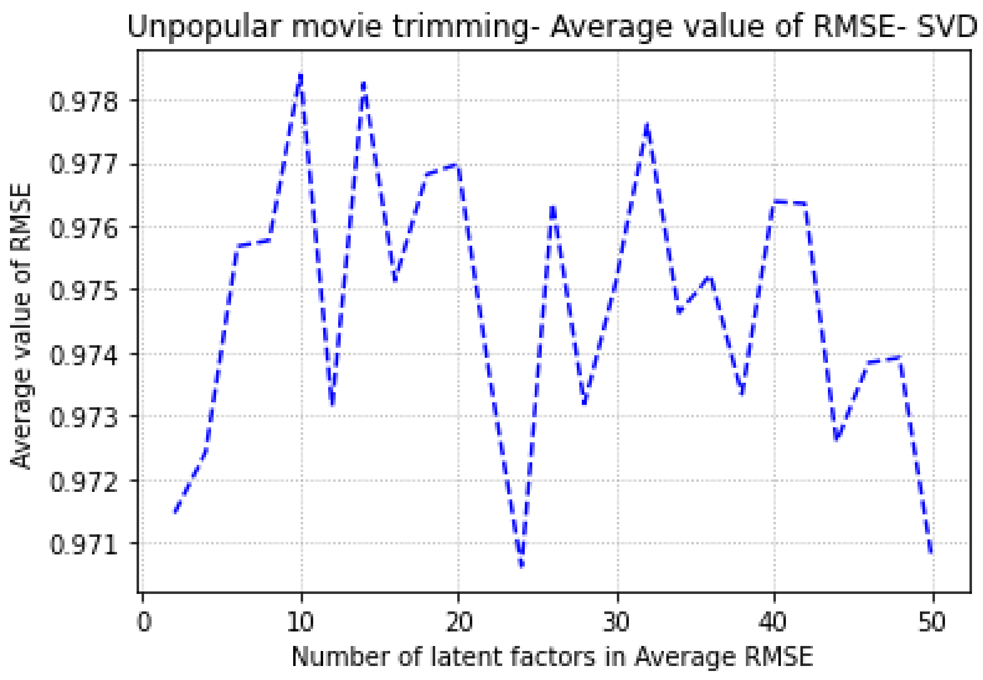


Figure 23: Average RMSE vs k for MF with bias (SVD) user-based CF on unpopular movie trimmed test set.

The minimum average RMSE for Unpopular movie trimming test set: **0.9706183651935959**. At k = 20, we observe **0.977**.
We notice that the above RMSE value is higher than that we obtained for the popular movie trimmed set and the original test set. We can claim that the model is not viable for the unpopular movie trimmed test set, because all of the movies left in the test dataset after trimming earn low user ratings. Because the predictor was trained on the full dataset, which includes popular movies, it has trouble accurately forecasting the ratings of "unusual" things because there aren't enough individuals in the area who have rated them.
We notice a jumpy pattern in the above figure. Rather than expressing the link between RMSE and different selections of k, the jumpy pattern in the above figure is created by distinct data splitting schemes rendered by different random states in cross-validation. Furthermore, the difference between RMSE is so small that the relevant curve is really a horizontal line if we partition the entire dataset in the same way every time.

- **High variance movie trimming:**
  In this trimming, we trim the test set to contain movies that have variance (of the

rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. If a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
We evaluated the performance of High variance movie trimmed test set using 10 fold cross validation. We used Pearson-correlation function as the similarity metric and swept k (number of latent factors) from 2 to 100 in step sizes of 2. For each k, we computed the average RMSE (Root Mean Square Error) across all 10 folds.

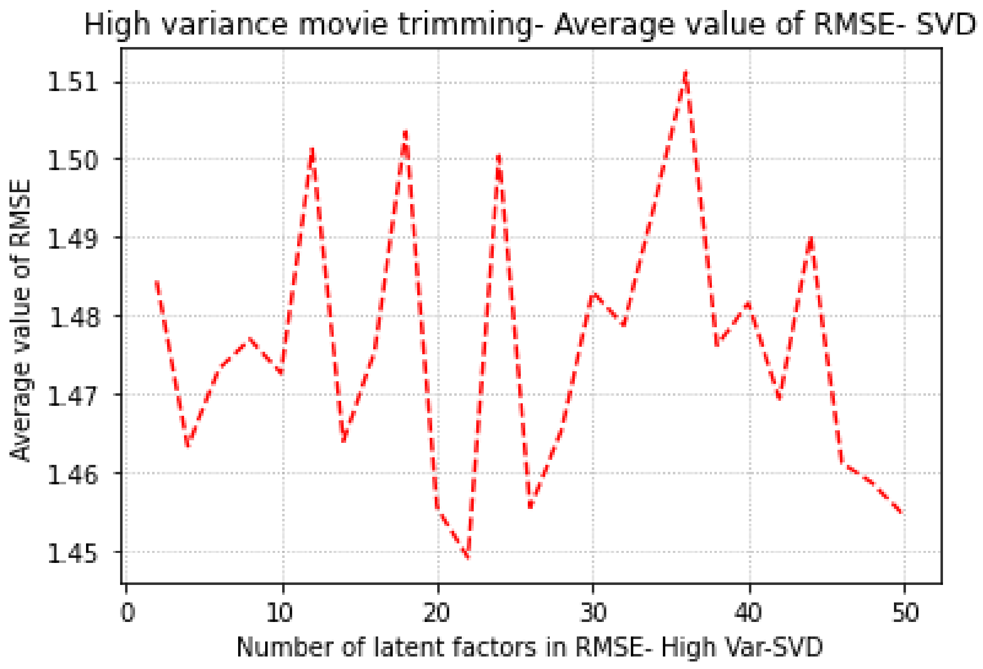


Figure 24: Average RMSE vs k for MF with bias (SVD) user-based CF on High variance movie trimmed test set.

The minimum average RMSE for High variance movie trimming test set: **1.4489840651169374**.
At k = 20, we observe **1.4563**.
The same scenario as in the shortened test set for an unpopular movie applies here. According to Question 1F, just a few films have a wide range of ratings. As a result, predicting on the test dataset after high variance movie trimming will result in the same issue as before.
The sorts of movies remaining in the testing dataset after high variance movie trimming are mostly contentious. Being controversial here implies that the film is somewhat popular (having a big number of ratings), yet the ratings supplied by various people disagree significantly (having large variance). As a result, predicting those films is difficult. That's why the plot has such erratic patterns.

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

We plot the ROC curve for the MF-biased Collaborative filters for the threshold values [2.5, 3, 3.5, 4]. We split the data into train and test, with a 90% and 10% split respectively. We used the k value found in the previous question i.e., k = 20. We also assumed that if the user likes the movie (rating is equal to or higher than a threshold), the label = 1.

The above ROC curve shows the plot for true positive rate Vs false positive rate. It is a measure of the relevance of the film recommended to the user.

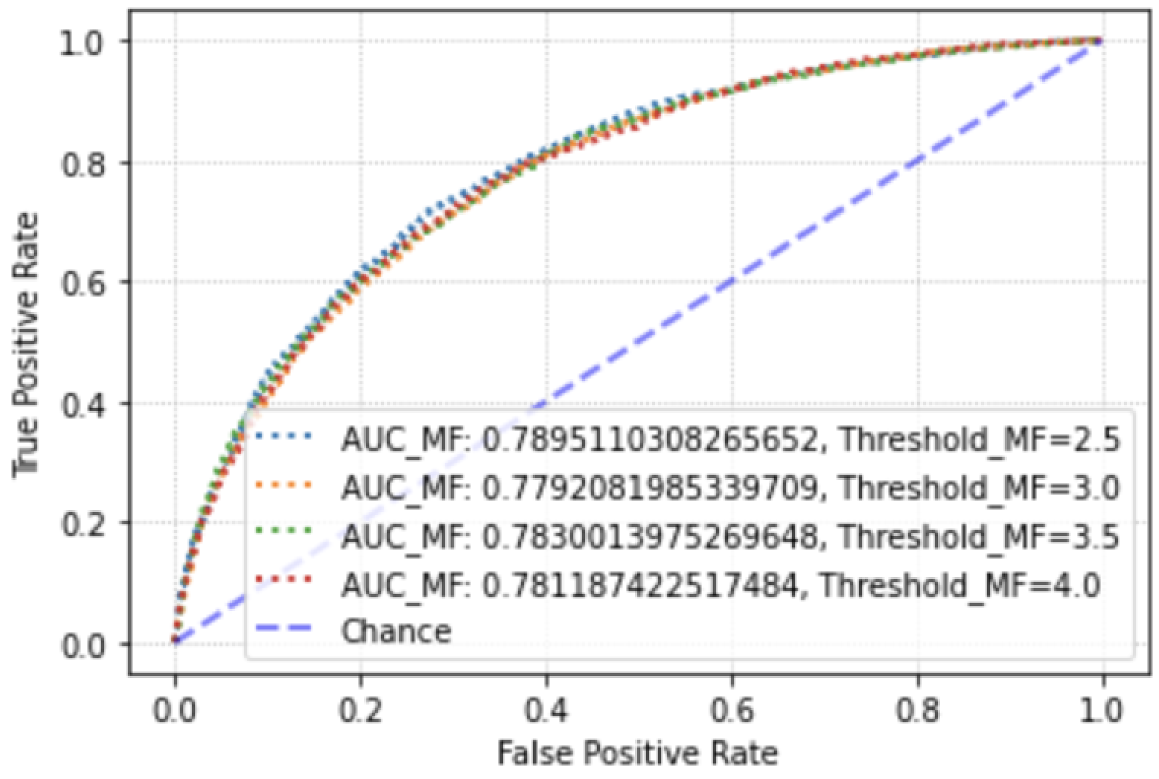


Figure 25: ROC curves for various thresholds for MF-biased user-based CF with k = 20 neighbors

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values.

- **Threshold 2.5:** 0.78951
- **Threshold 3:** 0.77920
- **Threshold 3.5:** 0.78300
- **Threshold 4** 0.78118

Threshold 2.5 renders the highest AUC score, which is 0.78951.

Question 11

Designing a Naive Collaborative Filter:
Design a naive collaborative filter to predict the ratings of the movies in the original dataset and evaluate it's performance using 10-fold cross validation. Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

We are asked to design a naive Collaborative Filter to predict the ratings of the movies in the given dataset. The naive CF simply returns the mean ratings of a user on past items as the rating for the new item. In simple words, the prediction function is nothing but the mean. Therefore there is no training involved and it is just direct prediction.
In the case of a naïve collaborative filter, we simply anticipate the average scores across all training data for each new user. As one may expect, this naïve model should provide a greater RMSE than both neighborhood-based and model-based collaborative filters. We determine the average RMSE of this novel prediction method utilizing surprise.AlgoBase to be **0.934689278702231**.
An important thing to note about the naive collaborative filter is that there is no notion of training. For training the model, split the dataset into 10 pairs of train set and test set and for each pair predict the ratings of the movies in the test set using the prediction function (no model fitting required). Then compute the RMSE for this fold and repeat the procedure for all the 10 folds. The average RMSE is computed by averaging the RMSE across all the 10 folds.

Performance on Test set subsets: For each of Popular, Unpopular and High-Variance test subsets:
– Design a naive collaborative filter to predict the ratings of the movies in the popular movie trimmed test set and evaluate it's performance using 10-fold cross validation.
– Compute the average RMSE by averaging the RMSE across all 10 folds. Report the average RMSE.

Here we are asked to design a naive collaborative filter to predict the ratings of the movie in the popular movie trimmed test set, unpopular movie trimmed test set and high variance movie test set. We evaluated it's performance using 10-fold cross validation. The RMSE was calculated across all the 10 folds, and the average RMSE value has been mentioned below.

Naive Collaborative Filter

- **Popular movie trimming:**
  In this trimming, we trim the test set to contain movies that have received more than 2 ratings. If a movie in the test set has received less than or equal to 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  The minimum average RMSE for Popular movie trimming test set: **0.9323153270544065**
  As observed previously, it is less than the without trimming scenario as all the unpopular outliers are removed.

- **Unpopular movie trimming:**
  In this trimming, we trim the test set to contain movies that have received less than or equal to 2 ratings. If a movie in the test set has received more than 2 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  The minimum average RMSE for Unpopular movie trimming test set: **0.971072554000641**
  As observed previously, it is more than the without trimming scenario as all the unpopular movies trimmed are outliers and hence perform poorly.

- **High variance movie trimming:**
  In this trimming, we trim the test set to contain movies that have variance (of the rating values received) of at least 2 and has received at least 5 ratings in the entire dataset. If a movie has variance less than 2 or has received less than 5 ratings in the entire dataset then we delete that movie from the test set and do not predict the rating of that movie using the trained filter.
  The minimum average RMSE for High Variance movie trimming test set: **1.481603271265438**
  The same scenario as in the shortened test set for an unpopular movie applies here. According to Question 1F, just a few films have a wide range of ratings. As a result, predicting on the test dataset after high variance movie trimming will result in the same issue as before.
  The sorts of movies remaining in the testing dataset after high variance movie trimming are mostly contentious. Being controversial here implies that the film is somewhat popular (having a big number of ratings), yet the ratings supplied by various people disagree significantly (having large variance). As a result, predicting those films is difficult. That's why the plot has such erratic patterns.

As observed earlier, RMSE of Popular test subset is slightly lesser than the entire dataset scenario. Also, the unpopular RMSE is slightly larger and that of high-variance the highest.

Question 12

Comparing the most performance models across architecture: Plot the best ROC curves (threshold = 3) for the k-NN, NMF, and MF with bias based collaborative filters in the same figure. Use the figure to compare the performance of the filters in predicting the ratings of the movies.
We plot and compare the ROC curves for the k-NN, NMF, MF-biased based Collaborative filter for the threshold value = 3. We split the data into train and test, with a 90% and 10% split respectively. We also assumed that if the user likes the movie (rating is equal to or higher than a threshold), the label = 1. We used the below k values (these k values were obtained from Questions 6, 8 and 10):

- k-NN user-based CF number of neighbors = 20
- NMF: latent factors = 18
- MF with bias (SVD): latent factors = 20

The above ROC curve shows the plot for true positive rate Vs false positive rate. It is a measure of the relevance of the film recommended to the user.

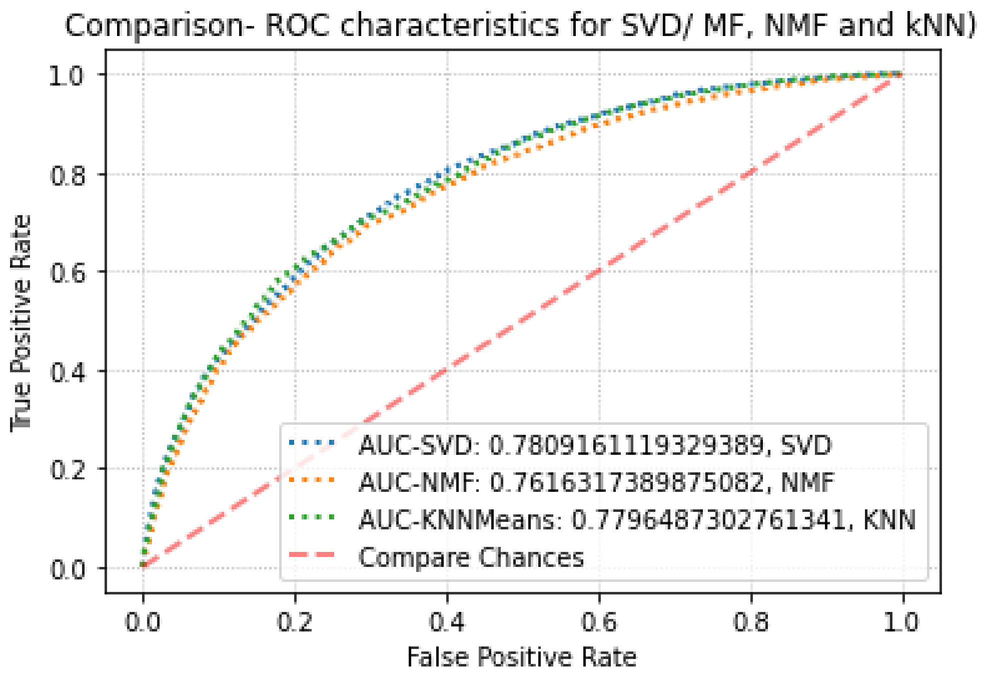


Figure 26: Comparison of ROC characteristics for MF with bias (SVD), NMF and kNN.

From the above figure, we can find the Area Under the Curve (AUC) for the threshold values.

- **k-NN:** 0.77964
- **NMF:** 0.76163
- **MF with bias (SVD):** 0.78091

From Figure 26, MF with bias seems to have the smoothest curve and has the largest area under the curve when threshold = 3. This means that MF with bias (SVD) is the best among the three collaborative filters at predicting the ratings of the movies, followed by

k-NN CF and then NMF. This means that MF with bias (SVD) is the optimal model for movie prediction. This makes sense, because MF with bias-based collaborative filter is more versatile than NMF-based collaborative filter because it introduces optimization variables for both user and movie bias. As a result, for those people who consistently offer an excessively high or poor rating, MF with bias-based collaborative filter might be beneficial.
In terms of movie rating prediction, however, the NMF-based collaborative filter performs the worst. This is a trade-off between forecast accuracy and model interpretability.
MF with bias (SVD) vs NMF:

- Because there are no limits on U and V, MF with bias can better represent higher-dimensional feature matrices, allowing for deep factorization with minimum information loss. NMF, on the other hand, requires U and V to be positive and has fewer ideal components in U and V than MF with bias.
- MF with bias generates a hierarchical and geometric basis that is ordered by relevance, resulting in embeddings with the most relevant qualities and traits in the ratings matrix further up in the hierarchy. As a result of the feature ordering, MF with bias embeddings are resistant to outliers and noise in the ratings. NMF, on the other hand, does not take geometry into account in the ratings matrix.
- MF with bias produces unique and deterministic embeddings, whereas NMF produces non-unique and stochastic embeddings with no assurances of convergence to the optimal U and V each time the function is applied.
- To lessen susceptibility to outliers and noise, MF with bias takes into account user and movie-specific bias information and normalizes it accordingly.

MF with bias (SVD) vs k-NN:

- The bias information is not modelled independently for each user or item by k-NN. As a result, it is more sensitive to outliers and products that are infrequently rated.
- k-NN infers directly on the sparse ratings matrix, resulting in low prediction performance in high-dimensional space (curse of dimensionality). This also reduces the recommender system's scalability. Because the ratings matrix is sparse, high-dimensional inference requires a significant quantity of training data to function well.
- When compared to latent-factor based models, k-NN is substantially less generalizable since it cannot identify semantic information and relationships within the user-item ratings matrix while being sensitive to infrequently rated items.

Question 13

Understanding Precision and Recall in the context of Recommender Systems: Precision and Recall are defined by the mathematical expressions given by equations 12 and 13 respectively. Please explain the meaning of precision and recall in your own words.

- **Precision:**
  Precision measures the percentage of correct recommended items (items liked by the user) among all the recommendations made to the user. It informs us the reliability of predictions rendered by some model. Low precision indicates high number of false positives.
  Precision at k is the proportion of recommended items in the top-k set that are relevant. Suppose that my precision at 10 in a top-10 recommendation problem is 80%. This means that 80% of the recommendation we make are relevant to the user.

- **Recall:**
  Recall measures the percentage of correct recommended items among the set of items liked by the user. This gives us a sense of how much of the real liked items (ground-truth positives) got retrieved by the model.
  Recall at k is the proportion of relevant items found in the top-k recommendations. Suppose that we computed recall at 10 and found it is 40% in our top-10 recommendation system. This means that 40% of the total number of the relevant items appear in the top-k results.

- Precision-recall curve can be used to evaluate the relevance of the ranked list. Before stating the expressions for precision and recall in the context of ranking, let's introduce some notation:
  S(t): The set of items of size t recommended to the user. In this recommended set, ignore (drop) the items for which we don't have a ground truth rating.
  G: The set of items liked by the user (ground-truth positives)

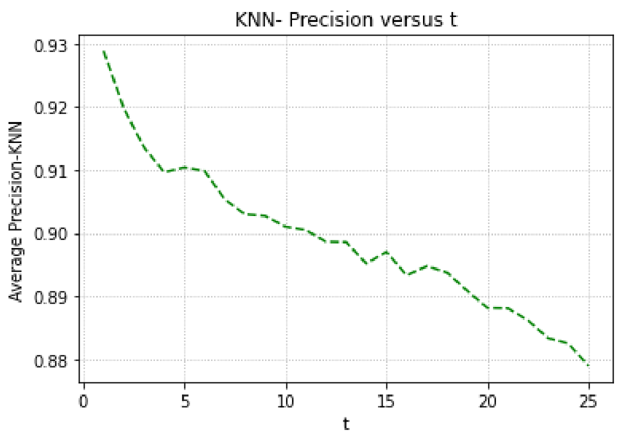
$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|}$$

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

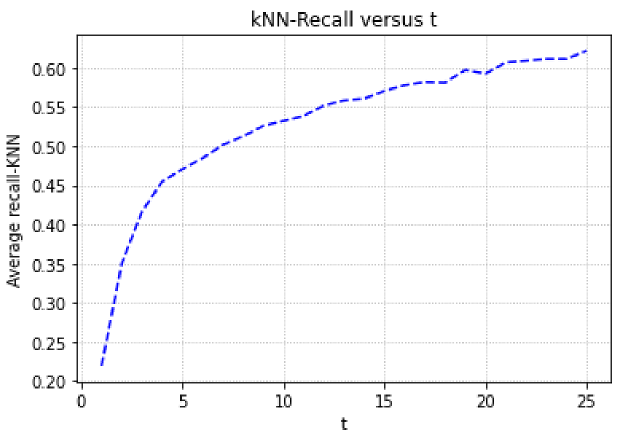
- Both precision and recall are functions of the size of the recommended list (t). Therefore, we can generate a precision-recall plot by varying t.

Question 14

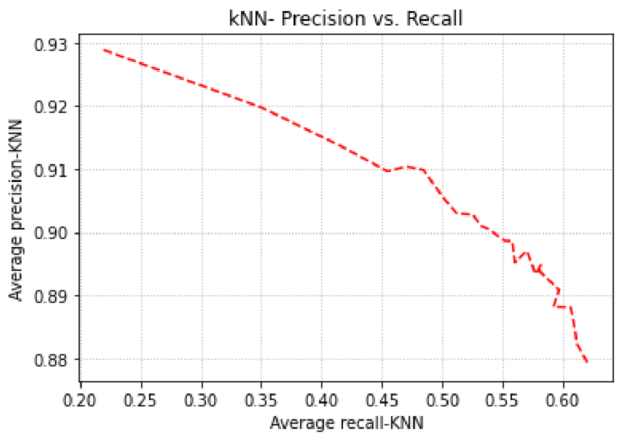
Comparing the precision-recall metrics for the different models: For each of the three architectures: – Plot average precision (Y-axis) against t (X-axis) for the ranking obtained using the model's predictions.
– Plot the average recall (Y-axis) against t (X-axis) and plot the average precision (Y-axis) against average recall (X-axis).
– Use the best k found in the previous parts and sweep t from 1 to 25 in step sizes of 1. For each plot, briefly comment on the shape of the plot.
- Plot the best precision-recall curves obtained for the three models (Naive, NMF, MF) in the same figure.
Use this figure to compare the relevance of the recommendation list generated using k-NN, NMF, and MF with bias predictions.
k-NN:
We show the plots of precision vs. t, recall vs. t and precision vs.accuracy curves for k-NN user-based CF with the best value of the number of neighbors found in Question 6, which is 20. t is the size of the set of items recommend to the user. In other words, we want to solve a ranking version of the recommender system. Below are the figures for the same:



(a) Average Precision vs t.



(b) Average Recall vs t.



(c) Average Precision vs Average Recall.

Figure 27: Precision-Recall metrics for k-NN Collaborative Filter

Remarks about the Graphs:

- The first subplot:
  Precision measures the percentage of correct recommended items (items liked by the user) among all the recommendations made to the user. When t grows, we see a

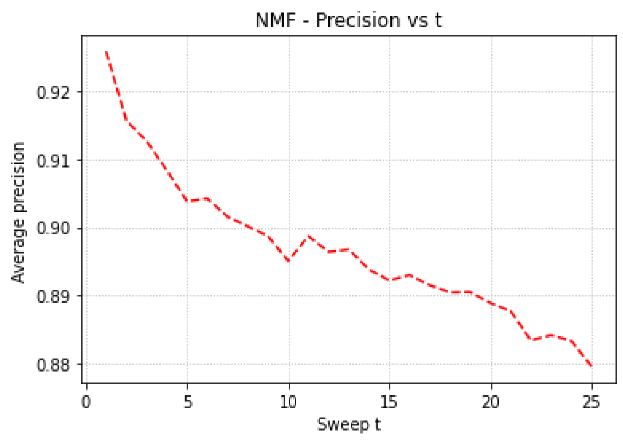
declining trend in precision. This makes logical sense, because if the recommendation system is asked to propose just one movie to the consumer, it will most likely hit the ground-truth positives by selecting the movie with the highest projected rating. As the number of recommended movies grows, say to 20, the 20th movie predicted may not be the one appreciated by the user, although having a projected rating that is reasonably near to the threshold of 3. Therefore, with larger t, it becomes more difficult to have all the t movies to be liked by the user.

- The second subplot:
  Recall measures the percentage of correct recommended items among the set of items liked by the user. Therefore, as we increase t and recommend more items, chances of more items in the user's preference list being recommended also increases. This explains why recall increases with increasing t. This can also be simply explained by looking at the mathematical formulation of recall: because the denominator, $|G|$, reflects the number of movies loved by the user, which is a constant, as t rises, the intersection between S(t) and G grows, resulting in a monotonic increase in recall. Also, there is a considerable improvement in recall in its early stages. This is because, say the $|G|$ could be 10, then recall will definitely be low for $t < 10$ as we are not even recommending 10 items. Therefore for small t, we observe sharp increase in recall. Once t is greater than $|G|$, we start observing that the increase in recall decreases.

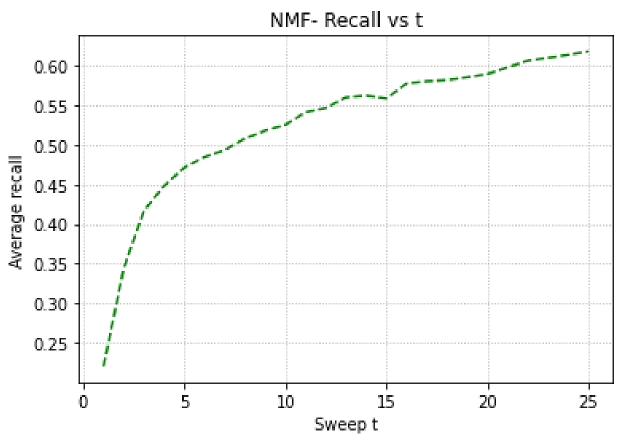
- The third subplot:
  Precision and recall have a negative connection. They have an inverse relationship. The reasons are simple: the link between precision and t is inverse, while the relationship between recall and t is positive. As a result, t bridges precision and recall and depicts the curve in the last subplot as a latent variable in between. Therefore, there is a trade off, we cannot maximize both precision and recall simultaneously.

NMF:

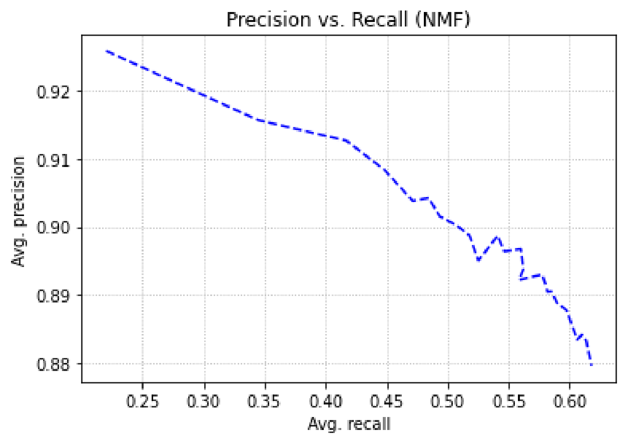
We show the plots of precision vs. t, recall vs. t and precision vs.accuracy curves for NMF user-based CF with the best value of the number of neighbors found in Question 8, which is 18. t is the size of the set of items recommend to the user. In other words, we want to solve a ranking version of the recommender system. Below are the figures for the same:



(a) Average Precision vs t.



(b) Average Recall vs t.



(c) Average Precision vs Average Recall.

Figure 28: Precision-Recall metrics for NMF Collaborative Filter

Remarks about the Graphs:

- The first subplot:
  Precision measures the percentage of correct recommended items (items liked by the user) among all the recommendations made to the user. When t grows, we see a declining trend in precision. This makes logical sense, because if the recommendation system is asked to propose just one movie to the consumer, it will most likely hit the ground-truth positives by selecting the movie with the highest projected rating. As the number of recommended movies grows, say to 20, the 20th movie predicted may not be the one appreciated by the user, although having a projected rating that is reasonably near to the threshold of 3. Therefore, with larger t, it becomes more difficult to have all the t movies to be liked by the user.

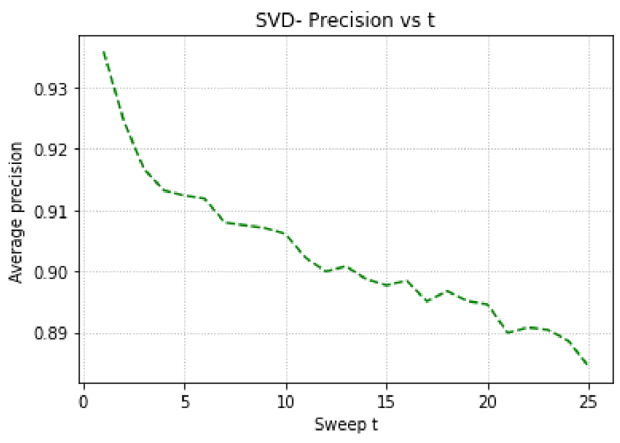
- The second subplot:
  Recall measures the percentage of correct recommended items among the set of items liked by the user. Therefore, as we increase t and recommend more items, chances of more items in the user's preference list being recommended also increases. This explains why recall increases with increasing t. This can also be simply explained by looking at the mathematical formulation of recall: because the denominator, $|G|$,

reflects the number of movies loved by the user, which is a constant, as t rises, the intersection between S(t) and G grows, resulting in a monotonic increase in recall. Also, there is a considerable improvement in recall in its early stages. This is because, say the $|G|$ could be 10, then recall will definitely be low for $t < 10$ as we are not even recommending 10 items. Therefore for small t, we observe sharp increase in recall. Once t is greater than $|G|$, we start observing that the increase in recall decreases.

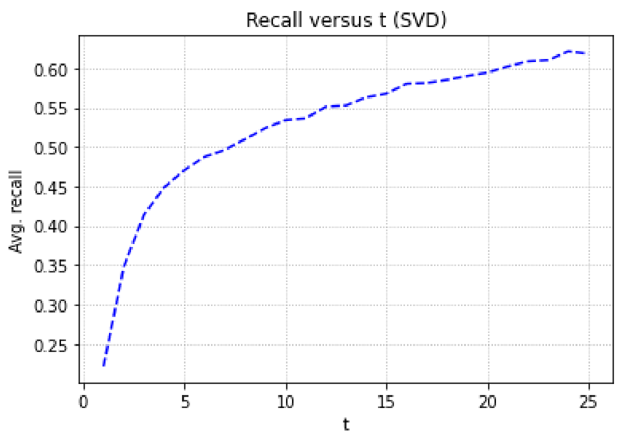
- The third subplot:
  Precision and recall have a negative connection. They have an inverse relationship. The reasons are simple: the link between precision and t is inverse, while the relationship between recall and t is positive. As a result, t bridges precision and recall and depicts the curve in the last subplot as a latent variable in between. Therefore, there is a trade off, we cannot maximize both precision and recall simultaneously.

MF with bias (SVD):

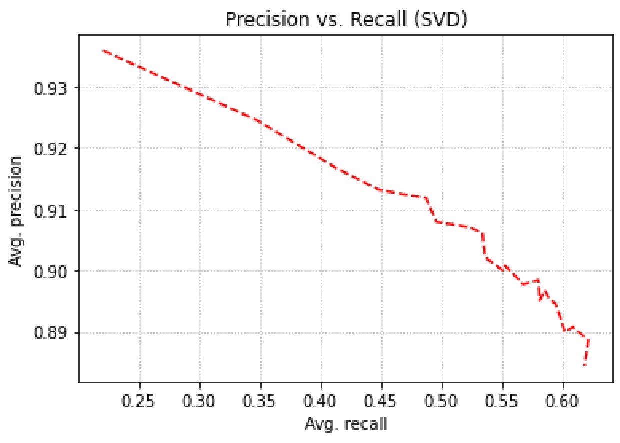
We show the plots of precision vs. t, recall vs. t and precision vs.accuracy curves for MF with bias (SVD) user-based CF with the best value of the number of neighbors found in Question 10, which is 20. t is the size of the set of items recommend to the user. In other words, we want to solve a ranking version of the recommender system. Below are the figures for the same:



(a) Average Precision vs t.



(b) Average Recall vs t.



(c) Average Precision vs Average Recall.

Figure 29: Precision-Recall metrics for MF Biased (SVD) Collaborative Filter

Remarks about the Graphs:

- The first subplot:
  Precision measures the percentage of correct recommended items (items liked by the user) among all the recommendations made to the user. When t grows, we see a declining trend in precision. This makes logical sense, because if the recommendation system is asked to propose just one movie to the consumer, it will most likely hit the ground-truth positives by selecting the movie with the highest projected rating. As the number of recommended movies grows, say to 20, the 20th movie predicted may not be the one appreciated by the user, although having a projected rating that is reasonably near to the threshold of 3. Therefore, with larger t, it becomes more difficult to have all the t movies to be liked by the user.

- The second subplot:
  Recall measures the percentage of correct recommended items among the set of items liked by the user. Therefore, as we increase t and recommend more items, chances of more items in the user's preference list being recommended also increases. This explains why recall increases with increasing t. This can also be simply explained

by looking at the mathematical formulation of recall: because the denominator, $|G|$, reflects the number of movies loved by the user, which is a constant, as t rises, the intersection between S(t) and G grows, resulting in a monotonic increase in recall. Also, there is a considerable improvement in recall in its early stages. This is because, say the $|G|$ could be 10, then recall will definitely be low for $t < 10$ as we are not even recommending 10 items. Therefore for small t, we observe sharp increase in recall. Once t is greater than $|G|$, we start observing that the increase in recall decreases.

- The third subplot:
  Precision and recall have a negative connection. They have an inverse relationship. The reasons are simple: the link between precision and t is inverse, while the relationship between recall and t is positive. As a result, t bridges precision and recall and depicts the curve in the last subplot as a latent variable in between. Therefore, there is a trade off, we cannot maximize both precision and recall simultaneously.

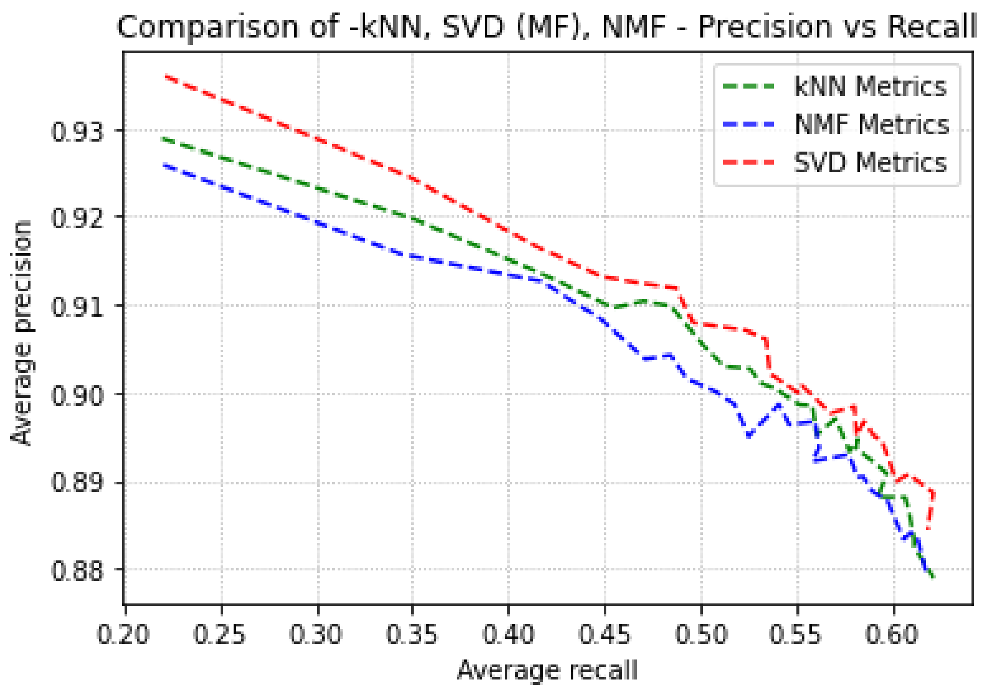


Figure 30: Comparison of k-NN, SVD (MF), NMF - Precision vs Recall.

The precision-recall curves for all three models show an inverse relationship between average precision and recall. However, MF with bias shows the best performance followed by kNN and then NMF. This means that average recall can be increased with only small decrease in the average precision for MF with bias.

Because precision and recall are separate metrics of prediction accuracy, we want them to be as near to one as feasible. If we wish to assess the relevance of the recommendation list created by different collaborative filters, we can simply fix precision (or recall) and compare the recall (or precision) for different models under the same precision value (or recall). Area under the precision-recall curve, like AUC for the ROC curve, may be used as a scalar metric to compare the performance of different recommendation systems. The precision-recall curves are depicted above.

The precision-recall curves in Figure 30 shows that MF with bias-based collaborative filter produces the best recommendation list of all three recommendation systems, whereas NMF-based collaborative filter performs the poorest. The conclusion is the same as that obtained using ROC curves. Similar reasoning for this outcome may be applied here as well.

MF with bias (SVD) vs NMF:

- Because there are no limits on U and V, MF with bias can better represent higher-dimensional feature matrices, allowing for deep factorization with minimum information loss. NMF, on the other hand, requires U and V to be positive and has fewer ideal components in U and V than MF with bias.

- MF with bias generates a hierarchical and geometric basis that is ordered by relevance, resulting in embeddings with the most relevant qualities and traits in the ratings matrix further up in the hierarchy. As a result of the feature ordering, MF with bias embeddings are resistant to outliers and noise in the ratings. NMF, on the other hand, does not take geometry into account in the ratings matrix.

- MF with bias produces unique and deterministic embeddings, whereas NMF produces non-unique and stochastic embeddings with no assurances of convergence to the optimal U and V each time the function is applied.

- To lessen susceptibility to outliers and noise, MF with bias takes into account user and movie-specific bias information and normalizes it accordingly.

MF with bias (SVD) vs k-NN:

- The bias information is not modelled independently for each user or item by k-NN. As a result, it is more sensitive to outliers and products that are infrequently rated.
- k-NN infers directly on the sparse ratings matrix, resulting in low prediction performance in high-dimensional space (curse of dimensionality). This also reduces the recommender system's scalability. Because the ratings matrix is sparse, high-dimensional inference requires a significant quantity of training data to function well.
- When compared to latent-factor based models, k-NN is substantially less generalizable since it cannot identify semantic information and relationships within the user-item ratings matrix while being sensitive to infrequently rated items.