Project 3: Recommender Systems

ECE 219 - Large Scale Data Mining
University of California, Los Angeles

Member 1:	Shruti Mohanty (705494615)
Member 2:	Kimaya Kulkarni (805528337)

1 Question 1

Explore the Dataset: In this question, we explore the structure of the data.

- **A.** Compute the sparsity of the movie rating dataset -
  In this question, we are asked to report the sparsity of the movie rating information for the Movie Lens dataset. The dataset is loaded from the Synthetic Movie Lens folder. The ratings.csv file is accessed to compute the ratings. The dataset contains 100836 ratings (in the range 0.5 to 5) for 9742 movies across 610 users.

  The formula for Sparsity is given by -

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

  where, Total number of possible ratings = Total number of users x Total number of movies. **The resulting sparsity is 0.0169996**. This indicates that the ratings matrix is sparse, where the rows denote users and the columns denote movies. Each element of this matrix denotes the rating of a movie provided by a user. The matrix is expected to be sparse because not every user could have rated every movie in the dataset.

- **B.** Plot a histogram showing the frequency of the rating values-
  We are asked to investigate the histogram showing the frequency of rating values. The figure below illustrates that.

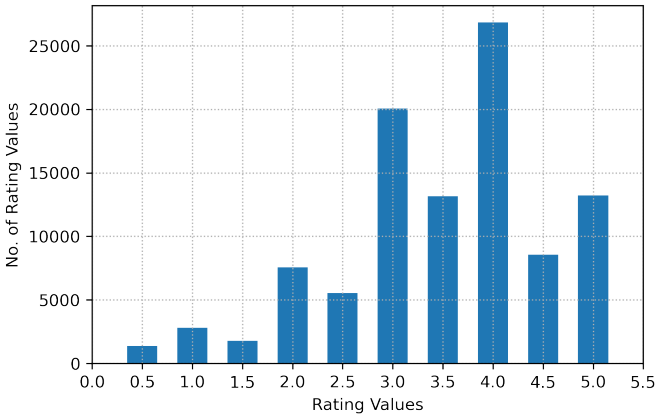

Figure 1:
Histogram showing the frequency of the rating values

Our observations from this histogram are that, most ratings by the users are between 3 and 5. The distribution of ratings is skewed to the left. This can be explained by the fact that most

viewers watch movies after looking at reviews from well-known ratings websites and sources as well as the hype generated around the movie. As a result, viewers usually only watch movies that they think they will like, and consequently, the user-ratings are higher as well. In addition, integer ratings (1, 2, 3, 4 and 5) have higher counts than fractional ratings.

- **C.** Plot the distribution of the number of ratings received among movies -
  In this question, we are asked to investigate the relationship between the number of ratings received and the movie. The figure below shows the plot of the number of ratings versus movie index, with the movie receiving the largest number of ratings indexed as 1. We have chosen to represent the movies in terms of their index rather than names for better readability on the plots.

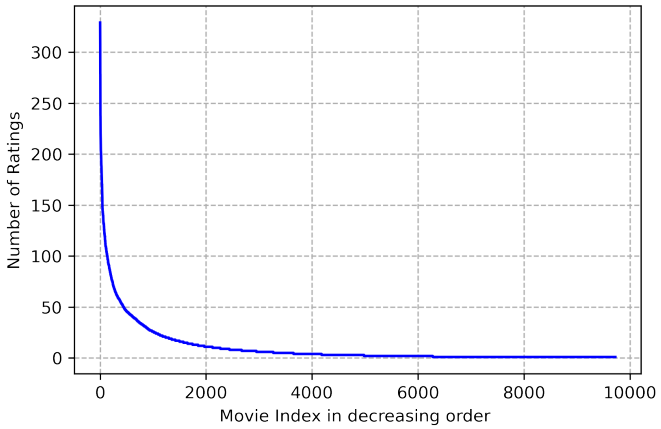

Figure 2:
Distribution of the number of ratings received among movies

From the above plot the sparsity of the ratings becomes clear. It is a monotonically decreasing function with only 500-700 movies receiving more than 50 unique user ratings. This fraction is very low compared to the total number of movies - approximately 9742.

- **D.** Plot the distribution of ratings among users
  In this question, we plot the number of ratings given by each user. The figure below shows this plot with the user providing the largest number of ratings indexed as 1.

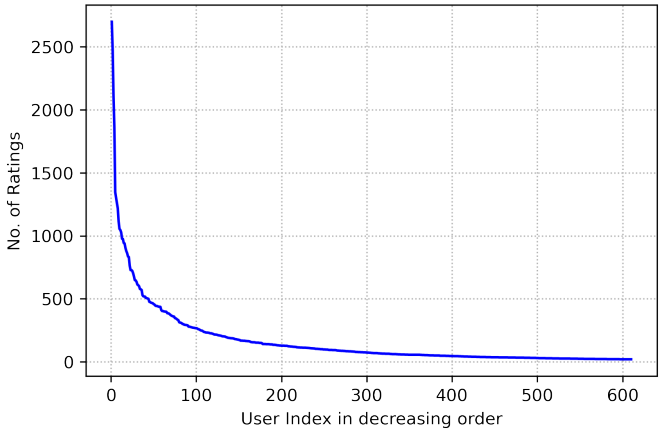

Figure 3:
Distribution of ratings among users

From this , we see that the curve is monotonically decreasing just like the previous question, with less than 50 users providing ratings to 500 movies or more (out of 9742). Similar to the

previous plot, this also explains why the ratings matrix is sparse, as a vast number of users do not provide enough unique ratings.

- **E.** Discuss the salient features of the distributions
  From the plots for C and D, we observe that curve is monotonically decreasing, with approximately 500 movies (out of 9742) receiving more than 50 unique user ratings. This indicates a vast majority of the movies did not receive enough unique user ratings, which explains the sparsity of the ratings matrix. Only a few movies are watched by a large audience because most viewers watch movies after looking at reviews from well-known ratings websites. From a machine learning perspective - we face the issue of " Curse of Dimensionality" as data sparsity is not coveted. As data moves into higher dimensions, the rapid increase in volume causes the data to become sparse in each dimension, causing the amount of data required for accurate classification and representation to increase exponentially for each dimension or feature.

  However, since most of the elements in the sparse representation are 0, these elements contribute no additional information for the model, resulting in a model with a large number of parameters that overfits on those movies with a higher number of user ratings, when compared to the movie with lesser ratings present.

  A common technique to address this problem is to use regularization to encourage generalization and prevent formation of ill-conditioned classifiers, leading to less complex models.

- **F.** Compute the variance of the rating values received by each movie
  In this question, we are asked to compute the variance of the rating values received by each movie. The figure below shows the histogram illustrating the variance in rating values.

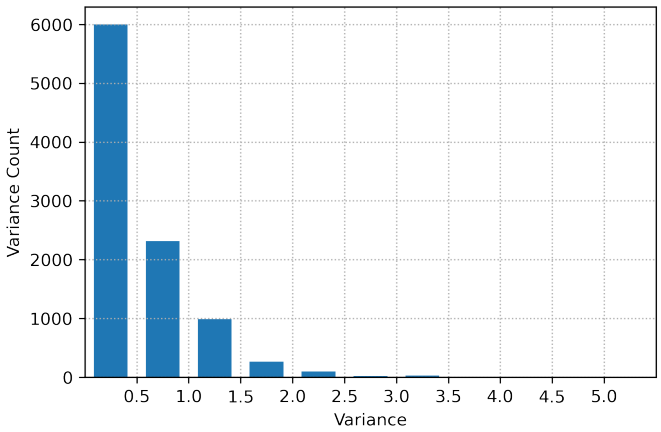

Figure 4:
Variance in rating values

From the above plot, we see that the distribution of variance is skewed to the right, with a large number of movies having a variance between 0 and 1.5. This means that the ratings provided to a certain movie are consistent across different users in the dataset. This is expected because most viewers watch movies after looking at reviews from well-known ratings websites resulting in similar ratings from all users for popular movies as we saw in plot for part B. This is also seen from the plot in Figure 1, which shows that most ratings are between 3 and 5.

2 Question 2

Understanding the Pearson Correlation Coefficient

- Write down the formula for μ_u in terms of I_u and r_{uk}

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

  Where I_u = Set of item indices for which ratings have been specified by user u, r_{uk} = Rating of user u for item k, and μ_u= Mean rating for user u computed using her specified ratings.

- In plain words, explain the meaning of $I_u \cap I_v$. Can $I_u \cap I_v = \emptyset$?

  I_u = Set of item indices for which ratings have been specified by user u, and I_v = Set of item indices for which ratings have been specified by user v.
  $I_u \cap I_v$ indicates the set of movie indices for which ratings have been specified for both user u and user v . This intersection can be null for the Synthetic Movie Lens dataset because as there may be some movies that have been rated by user u but not user v and vice versa. We will address this problem and build recommender systems for the missing rating. This is also the reason why the ratings matrix is sparse in nature.

3 Question 3

Understanding the Prediction function

This centering is done to reduce the impact of users who either rate all items highly or rate all items poorly. Centering the raw ratings around the mean ratings of the users reduces the user-specific bias in the ratings, as well as variance caused by polar or critical ratings by certain users. These ratings are called outliers. Some users might provide ratings at the higher or lower end of the rating spectrum between 4 and 5 or between 1 and 3 while other users might mix and match ratings within the entire range of the spectrum between 1 and 5. Mean centering helps remove these traits and outliers to make the data less noisy. Since we want to find the interaction between user ratings in the prediction function, mean-centering helps remove multicollinearity between the predictor variables, making it easier to find the significance of individual user ratings.

4 Question 4

Design a k-NN collaborative filter

In this question we are asked to 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).

A Collaborative filter (CF) uses past ratings across users for similar items to predict items that a user might be interested in by exploiting the correlation of ratings across users or items from a sparse matrix of user ratings. k-NN is a neighborhood-based CF model, which uses behavior of similar users or behavior of users on similar items to recommend new items to users. There are two types of neighborhood-based CF model - user based and item based. We use user-based CF using k-NN in this question. A user based model exploits similarity across users to predict ratings on new items.

The steps employed by us to use a k-NN collaborative filter are -

- To predict the rating of an item by a user, find the top k similar users to the target user who have also rated the item.
- Use the weighted average of the ratings of the k similar users to predict ratings for the target user on target item. We have defined the prediction functions in Q2 and Q3. r_{uk} = Rating of user u for item k, and μ_u= Mean rating for user u computed using her specified ratings. Mean centering is used in this calculation.
- To compute the similarity used in calculating r_{uk}, the Pearson similarity metric has been used.

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

The centered cosine similarity metric takes into account the varying baselines (user-specific bias, traits and polar ratings) and attempts to normalize these differences. On the other hand, euclidean distances converge to a constant value between all sample points in higher dimensions for sparse ratings matrix, which can degrade recommendation performance. Hence euclidean distances are not used in this project.

We use the Surprise-Scikit library's KNNWithMeans() for this question to implement the K-NN user-based collaborative filter on the ratings.csv file, while using the same library's cross_validate() function to perform 10-fold cross-validation. k was swept between 2 and 100. The figure below shows the average prediction root mean squared error (RMSE) and mean absolute error (MAE) for various number of neighbors (top k users) with Pearson similarity metric. This was selected by including sim_options='name':'pearson' in the code.

From the plots we notice that after a particular k value the average error saturates. As the number of users in the neighborhood increases, the prediction error drops monotonically until a critical point, after which increasing k does not result in a significant decrease in prediction error. With more neighbors, the error decreases up till saturation as the k-NN Collaborative filter has more users within the vicinity to compare the ratings with. The comparison parameter used is the Pearson similarity metric.

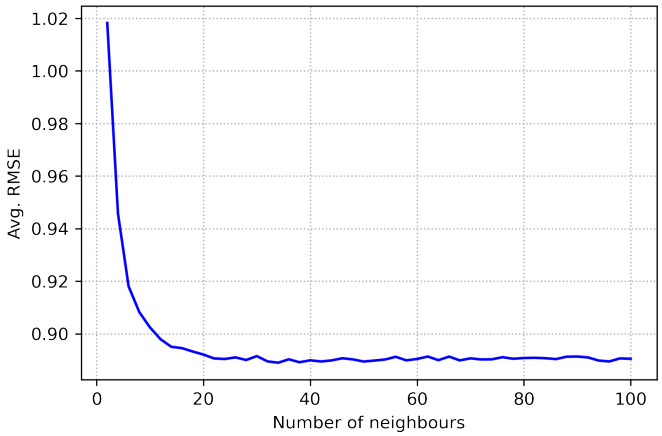

Figure 5:
Average RMSE against k for k-NN

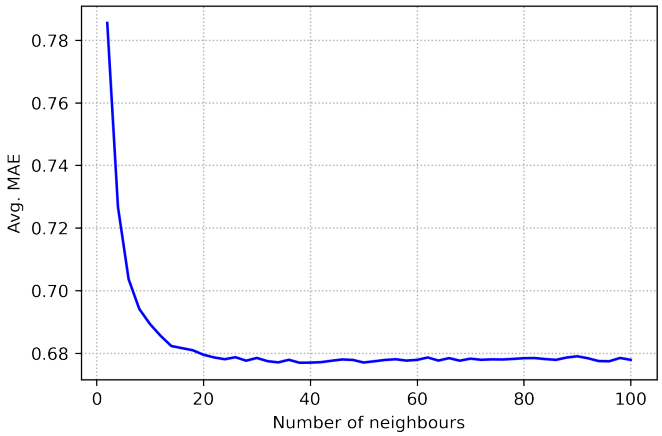

Figure 6:
Average MAE against k for k-NN

5 Question 5

Find the "minimum" k.

In this question, we are asked to find the minimum value of k, beyond which the errors for k-NN user-based collaborative filter level out. From the plots in Question 4, we see that this occurs at **k = 20**, with the steady-state average **RMSE = 0.892** and steady-state average **MAE = 0.679**. The value of k has been eyeballed as the minimum. Beyond this value additional Ks don't make a significant difference in the reported error values.

6 Question 6

Design a k-NN collaborative filter for 3 subsets in the testset

The ways in which the test set can be divided -

- Popular movie trimming - We trim the test set to contain movies that have received more than 2 ratings.
- Unpopular movie trimming - We trim the test set to contain movies that have received less than or equal to 2 ratings.
- High variance movie 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.

We want to design a k-NN collaborative filter to predict the ratings of the movies in the test subset (i.e Popular, Unpopular or High-Variance) and evaluate each of the three models' performance using 10-fold cross validation. These steps were used -

- For each value of k, split the dataset into 10 pairs of training and test sets. (trainset 1, testset 1), (trainset 2, testset 2), . . . , (trainset 10, testset 10)
- For each pair of the (trainset, testset) - Train the collaborative filter on the train set, Write a trimming function that takes as input the test set and outputs a trimmed test set, Predict the ratings of the movies in the trimmed test set using the trained collaborative filter, Compute the RMSE of the predictions in the trimmed test set
- Compute the average RMSE by averaging across all the 10 folds.
- For the ROC plotting, split the dataset into 90% for training and 10% for testing.

Results of the k-NN CF for 3 subsets of the testset -

- **A. Popular movie trimmed set** -

  In this question, we are asked to design a k-NN user-based CF, with predictions being made on the popular movie trimmed test set, which contains only those movies that have received at least 2 ratings. We use the KFold() function in Surprise-Scikit library to split the dataset into 10 pairs of train and test folds and remove those movies from the test set that have less than 2 ratings in the entire dataset. We test the CF with Pearson similarity metric for various number of neighbors, training on the training folds and testing on the trimmed set. We calculate the RMSE for each test fold and average. The figure below shows the average prediction RMSE for various number of neighbors (users).

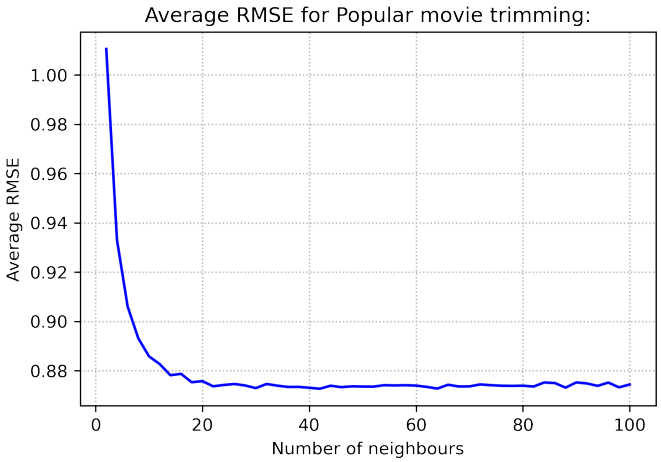

Figure 7:
Average RMSE against k for k-NN with popular movie trimmed set

The minimum average RMSE for k-NN user-based CF on popular movie trimmed test set is **0.872**. From the above figure, we see that the trend of monotonically decreasing prediction error continues similar to the figure from Question 4, but with lower prediction error compared to testing on the entire dataset (0.892) . This is expected because trimming out movies with fewer ratings removes the outliers from our dataset, which can be difficult to evaluate by the prediction function due to lack of enough ratings or neighbors (users) for those movies, as our rankings matrix is sparse in nature.

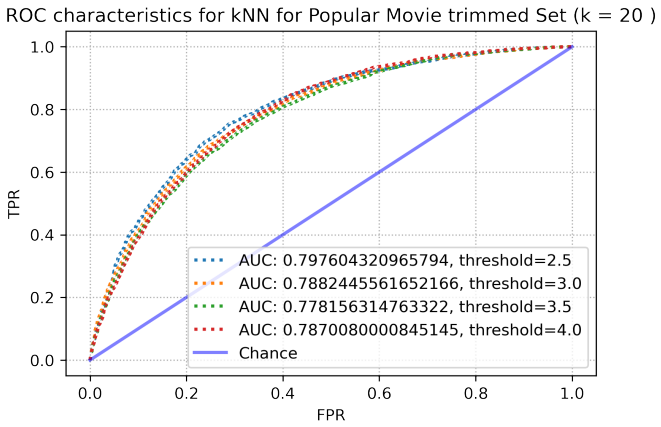

Figure 8:
ROC curves for k-NN with popular movie trimmed set

We are also asked to plot the ROC curves for k-NN user-based CF using k =20, for thresholds [2.5, 3, 3.5, 4]. The ROC curve shows the plot for true-positive (TP) rate vs false-positive (FP) rate. For recommender systems, it is a measure of the relevance of the items recommended to the user. We used train_test_split from Surprise-Scikit to split the dataset into 90% training and 10% testing, while using roc_curve() and auc() from SciKit-Learn to get the TPR, FPR and area-under-curve (AUC) for each threshold. The above figure shows the ROC plots for the 4 thresholds along with their AUC values reported.

The AUC for each threshold are as follows for the popular movie trimmed set -

- Threshold: 2.5, AUC: 0.7940
- Threshold: 3.0, AUC: 0.7904

– Threshold: 3.5, AUC: 0.7853
– Threshold: 4.0, AUC: 0.7766

We see that as threshold increases, the AUC starts to decrease for this set.

- **B. Unpopular movie trimmed set**

  In this question, we are asked to design a k-NN user-based CF, with predictions being made on the unpopular movie trimmed test set, which contains only those movies that have received less than or equal to 2 ratings in the entire dataset. We calculate the RMSE for each test fold and average. The figure below shows the average prediction RMSE for various number of neighbors (users).

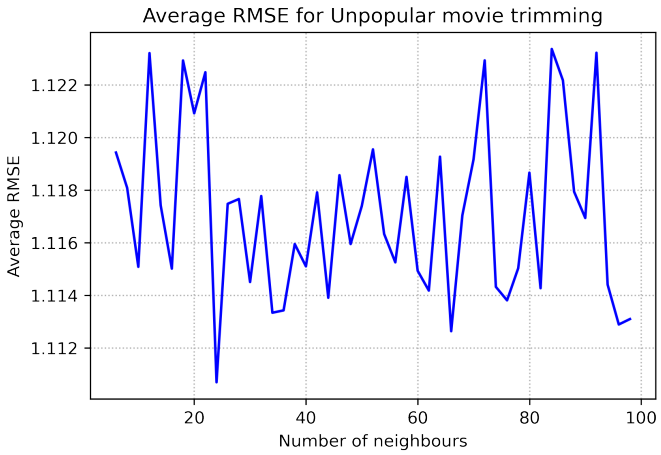

Figure 9:
Average RMSE against k for k-NN with unpopular movie trimmed set

The minimum average RMSE for k-NN user-based CF on unpopular movie trimmed test set is **1.110**. This is a lot higher than the minimum average RMSE for popular movie trimmed test set and the untrimmed set. The RMSE vs k curve becomes non-monotonic in nature with erratic jumps across neighbors. This is expected because the test set is now full of outliers containing only those movies with insufficient number of ratings. Since the predictor was trained on the entire dataset which also contains popular movies, the predictor has difficulty in correctly estimating the ratings of the rare items as it does not find enough users within the neighborhood who have rated the movies. As a result, the error is now effectively high and varying k does not result in any predictable change in average RMSE.

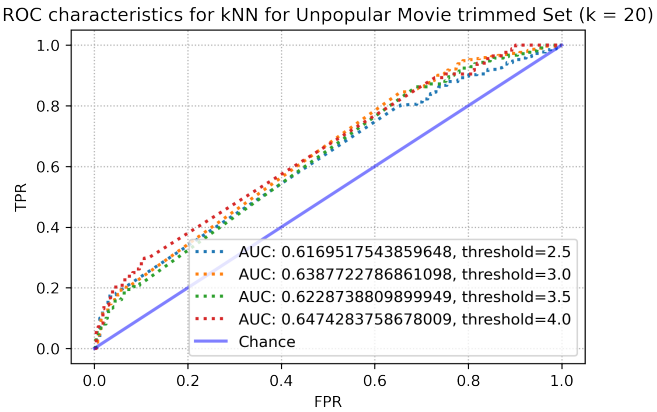

Figure 10:
ROC curves for k-NN with unpopular movie trimmed set

We are also asked to plot the ROC curves for k-NN user-based CF using k =20, for thresholds [2.5, 3, 3.5, 4]. The above figure shows the ROC plots for the 4 thresholds along with their AUC values reported.

The AUC for each threshold are as follows for the unpopular movie trimmed set -

- Threshold: 2.5, AUC: 0.616
- Threshold: 3.0, AUC: 0.638
- Threshold: 3.5, AUC: 0.622
- Threshold: 4.0, AUC: 0.647

The AUC values for unpoular trimmed set are lesser than that for Popular movie trimmed set. That is expected as the RMSE errors were also higher for unpopular movie trimmed set.

- **C. High variance movie trimmed set**

  In this question, we are asked to design a k-NN user-based CF, with predictions being made on the high-variance movie trimmed test set, which contains only those movies that either has a variance of at least 2 or has at least 5 ratings in the entire dataset. The figure below shows the average prediction RMSE for various number of neighbors (users).

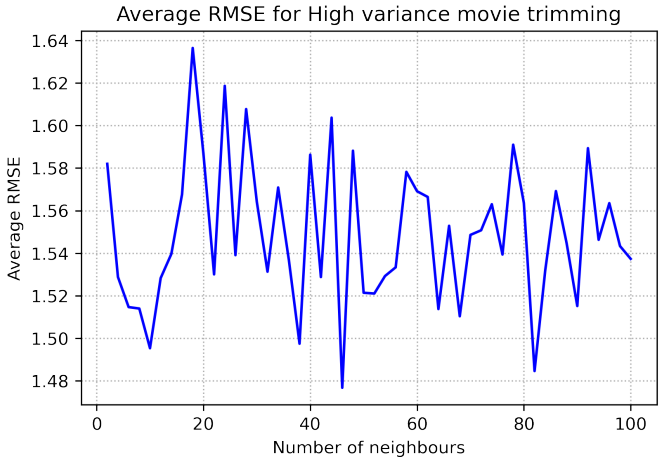

Figure 11:
Average RMSE against k for k-NN with high variance movie trimmed set

The minimum average RMSE for k-NN user-based CF on high-variance movie trimmed test set is **1.476**. This is worse than the prediction error in both the popular and unpopular movie trimmed test set, and the curve shows non-monotonic behavior similar to what was observed in the previous plot. There are much more outliers present in this test subset. With the test set containing only movies with erratic ratings, the predictor (which was trained to generalize on the entire training set) provides softer ratings for the movies with polar ratings, which results in a large prediction error that is effectively decoupled from the number of neighbors. So the value of k doesnt have much effect on the plot, hence explaining its highly monotonic nature.

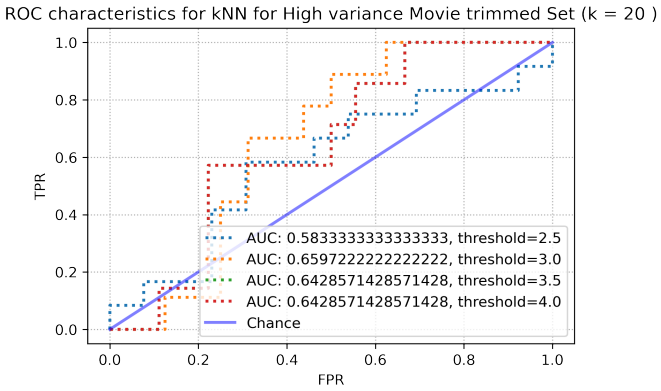

Figure 12:
ROC curves for k-NN with high variance movie trimmed set

We are also asked to plot the ROC curves for k-NN user-based CF using k =20, for thresholds [2.5, 3, 3.5, 4]. The above figure shows the ROC plots for the 4 thresholds along with their AUC values reported.

The AUC for each threshold are as follows for the unpopular movie trimmed set -

- Threshold: 2.5, AUC: 0.583
- Threshold: 3.0, AUC: 0.659
- Threshold: 3.5, AUC: 0.642
- Threshold: 4.0, AUC: 0.642

The AUC values for high variance trimmed set are lesser than that for popular movie trimmed set and unpopular trimmed set. This is because of the reasons explained above where the test set has a lot of outliers present, on which the model can't generalize well.

7 Question 7

Understanding the NMF cost function

The optimization problem is not jointly convex for the user latent space - U and item embedding space- V because of the existence of multiple local minima in the objective function gradient plane.

This function is a non constant function with more than 1 global minima. The Hessian matrix for this equation is not positive definite. This is because the matrix factorization model predicts ratings using the product of U and V, which does not satisfy the property of convexity as the objective function being considered here which is permutation and rotation invariant.

The optimization problem can be solved using alternating least-squares (ALS), keeping U fixed and solving for V and vice versa in the next step. That makes it a 2 step iterative process. Least squares solution is guaranteed to converge to a local minima depending on the initial values of U and V. For fixed U, the least-squares formulation of the objective function is given by the following equation. This equation doen't take into consideration the effect of regularization, and R is the ratings matrix.

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

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

This is the solution to any problem solved using least squares without the regularisation λ.

8 Question 8

1. In this question we design a non-negative matrix factorization (NNMF/NMF) CF to predict the ratings of the movies in the MovieLens dataset. We also explore the relationship between number of latent factors and the prediction error. NNMF-CF is a model-based CF, more specifically, a latent factor-based CF. In general Model-based CF are more robust against noise and sparsity in the ratings matrix. It uses deep-learning, clustering or matrix completion (latent-factor based CF) to predict user rating on new items without using the sparse ratings matrix R every time to make a prediction. This results in fast, scalable and generalizable recommendation system. In matrix factorization-based CF, R is expressed in terms of user latent space, U and item embedding space, V (which are low dimensional hidden factors for items and users), which are not sparse. Matrix decomposition, which is a simple embedding model, can be reformulated as an optimization problem with the following objective function:

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

   The last two terms in above equation act as regularizer. Regulizer is used encourage generalization and prevent overfitting. The optimization problem with W is called weighted matrix factorization. In NNMF, we set constraints on the U and v such that both will always be greater than equal to zero.
   We use the Surprise-Scikit library's NMF() for this question to implement the NNMF CF on the movie rating information. The following figure shows the average prediction root RMSE and MAE for various number of latent factors.

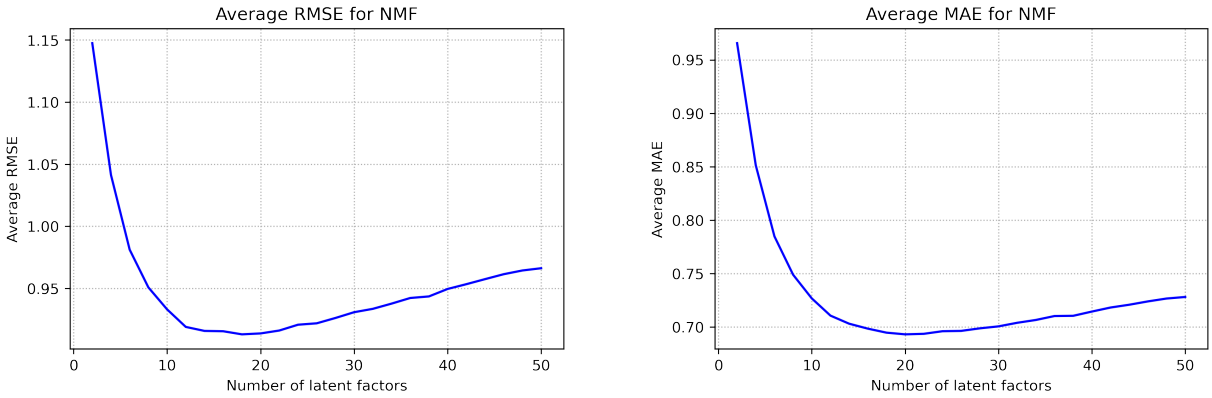

Figure 13: (Left) Average RMSE vs. k for NNMF CF (Right) Average MAE vs. k for NNMF CF

From the above figure it can be seen that as the number of latent factors increase, the prediction error decreases sharply till a critical point. After this critical point the prediction error starts to increase linearly. If the number of principal components (latent factors) is more, then the amount of semantic and complex information and structure in the data available for making predictions increases. This explains the initial improvement in prediction error.
However, very large values of k indicate a high-dimensional and noisy ratings matrix, which degrades matrix completion performance. This is because in higher dimensions, the rapid increase in volume causes the factorized matrix to approach the original sparse matrix. Making predictions on the sparse matrix results in degradation of prediction error, which is explained by the rise in error in the later portions of the RMSE and MAE curves. In addition, NMF only allows positive entries in the reduced-rank embeddings, providing a shallow factorization with high information loss, and causing the decomposition depth of NMF to diminish in high-dimensions. Furthermore, NMF does not consider the geometry in the feature space basisand is

non-uniqueand stochastic, with no guarantees of convergence to the optimal embeddings each time the function is called.

2. In this section, we find the optimum value of the number of latent factors for which the RMSE and MAE errors are minimal for NNMF CF. Judging from the curves in the above figure, we see that this occurs at k = 18 for RMSE, with the minimum average RMSE = 0.913044 and k = 20 for MAE, with the minimum average MAE = 0.692898. We stick to k = 18 after this point onwards. There are 19 genres in the MovieLens dataset, which is very close to the optimal value of k (which is 18). Note that the average of 18 and 20 is 19, which is exactly the number of movie genres.

3. In this section, we design an NNMF CF with predictions being made on the popular movie trimmed test set. The following figure shows the average prediction RMSE for various number of latent factors.

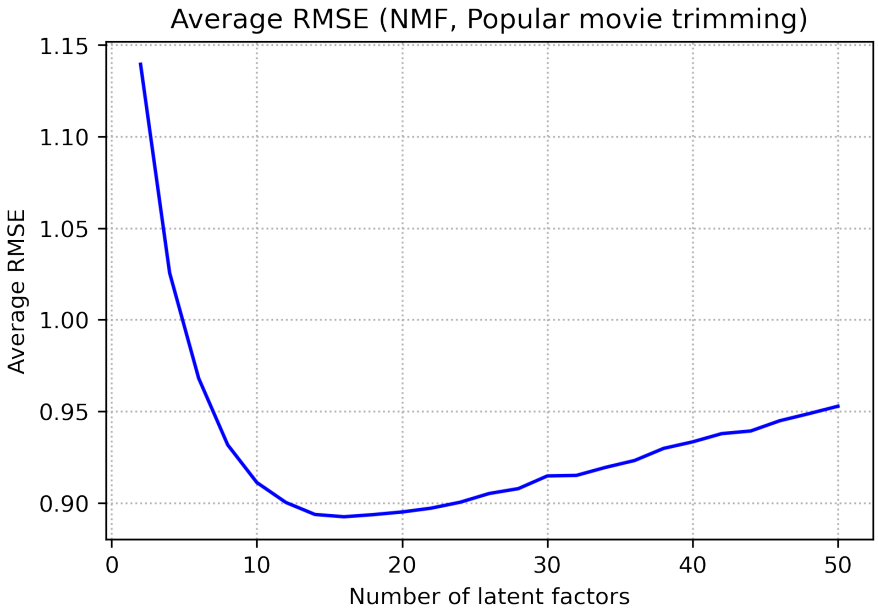

Figure 14: Average RMSE vs k for NNMF CF on popular movie trimmed test set

The minimum average RMSE for NNMF CF on the popular movie trimmed test set is 0.8924. We have provided an explanation of the initially decreasing and then rising prediction error characteristics of the curve in the previous section.
Moreover, we design an NNMF CF with predictions being made on the unpopular movie trimmed test set. Following figure shows the average prediction RMSE for various number of latent factors

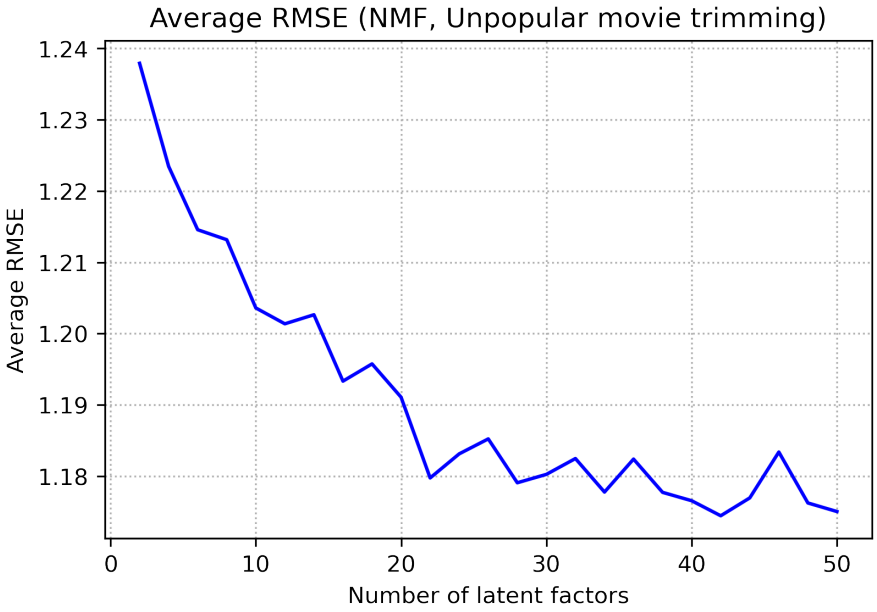

Figure 15: Average RMSE vs k for NNMF CF on unpopular movie trimmed test set

The minimum average RMSE for NNMF CF on the unpopular movie trimmed test set is 1.1744. This is larger than the minimum average RMSE on the popular movie trimmed test set. This is expected because the test set is now full of outliers containing only those movies with insufficient number of ratings. Since the predictor was trained on the entire dataset which also contains popular movies, the predictor has difficulty in correctly estimating the ratings of the uncommon items as it did not have enough training data to correctly generate embeddings for those particular movies during training time. This also explains the erratic rise in the RMSE in the latter portion of the graph.
We alsp design an NNMF CF with predictions being made on the high-variance movie trimmed test set. Figure 13 shows the average prediction RMSE for various number of latent factors.

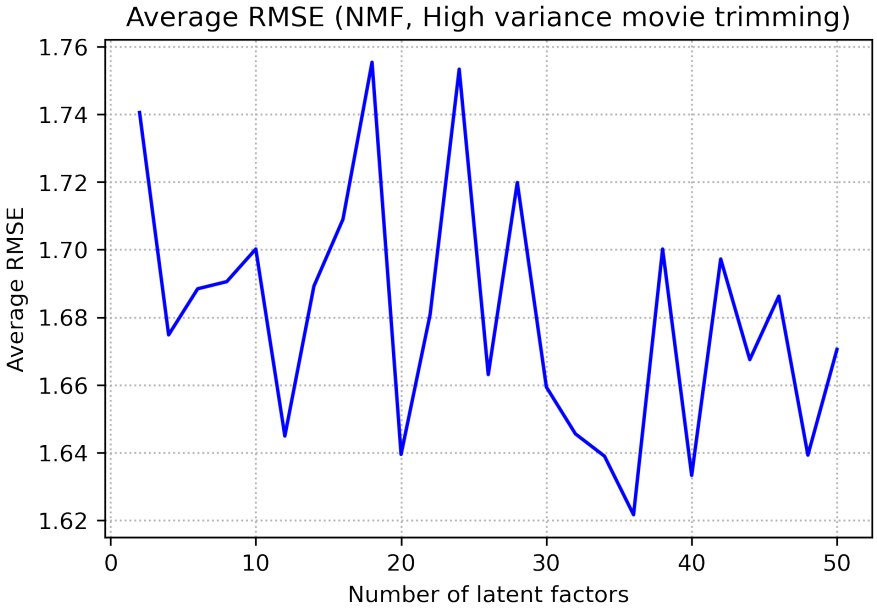

Figure 16: Average RMSE vs k for NNMF CF on high-variance movie trimmed test set

The minimum average RMSE for NNMF CF on the high-variance movie trimmed test set is 1.6216. With the test set containing only movies with erratic ratings, the predictor (which was trained to generate generalizable embeddings on the entire training set) provides softer ratings for the movies with polar ratings, which results in a large prediction error that is effectively decoupled from the number of latent factors. Since the ratings are polar and vary wildly, the average ratings of each movie are sensitive to outliers and cannot be properly converted to low-dimensional embeddings or imputed. This results in worse average RMSE compared to popular and unpopular movie trimmed test set.

4. In this section, we are asked to plot the ROC curves for NNMF CF for best value of k found in previous question, for thresholds [2.5, 3, 3.5, 4]. The following figure shows the ROC curves for the four thresholds for NNMF CF with 18 latent factors.

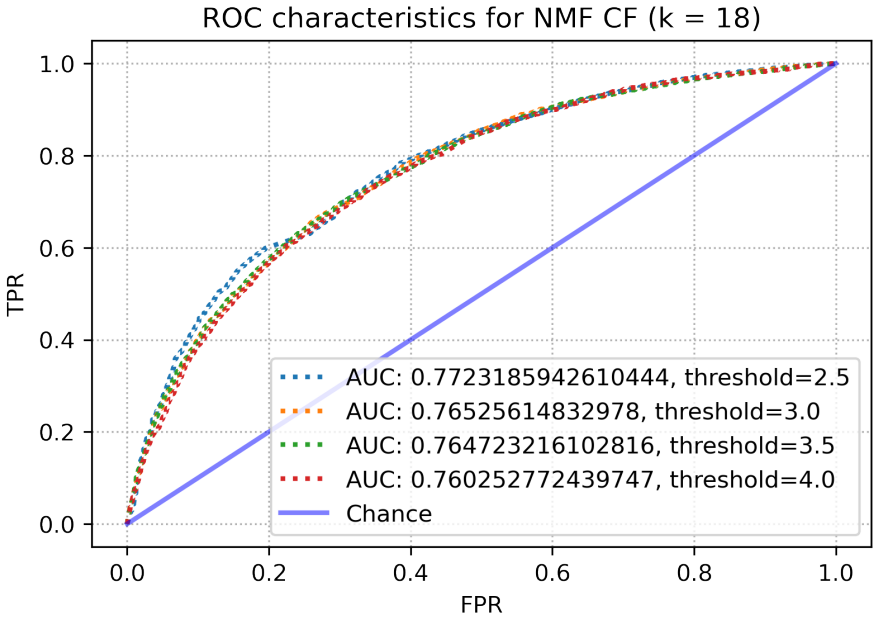

Figure 17: ROC curves for various thresholds for NNMF CF with 18 latent factors

The AUC for each threshold are as follows:

Threshold: 2.5, AUC: 0.772318
Threshold: 3.0, AUC: 0.765256
Threshold: 3.5, AUC: 0.764723
Threshold: 4.0, AUC: 0.760252

It can be observed that as the threshold increases, the AUC decreases monotonically for NNMF CF.

9 Question 9

The connection between the latent factors and the movie genres to check the interpretability of NNMF CF was investigated in this question. V matrix was obtained using nmf.qi command, where $nmf = NMF(n_factors = 20, n_epochs = 50, verbose = False)$.
k was set to 20 as metioned in the questio. Following this we sorted the rows of V in descending order and printed out the top 10 movies for random columns of V. The results are as follows

Column number of V: 1
Comedy—Drama—Sci-Fi—War
Drama
Drama—Mystery—Thriller
Comedy—Romance
Action—Comedy
Adventure—Animation—Children—Drama—Fantasy
Drama—Romance
Crime—Drama—Thriller
Adventure—Comedy
Comedy—Drama—Romance

Column number of V: 3
Comedy—Musical—Romance
Drama
Action—Adventure—Drama—Western
Drama
Comedy
Comedy—Crime—Drama—Mystery—Romance
Comedy—Romance
Comedy—Horror—Thriller
Horror—Sci-Fi—Thriller
Drama

Column number of V: 5
Action—Crime—Thriller
Comedy
Comedy—Drama—Romance
Adventure—Children—Drama
Adventure—Animation—Children—Drama—Fantasy
Children—Comedy—Drama
Crime—Drama—Mystery—Thriller
Comedy—Drama
Animation—Children—Comedy—Drama
Drama

Column number of V: 7
Romance—Western
Drama
Action—Adventure—Sci-Fi—Thriller
Action—Drama
Comedy—Romance
Comedy—Fantasy—Romance
Action—Romance

Comedy—Romance
Action—Comedy—Crime
Action—Adventure—Fantasy—Sci-Fi

Column number of V: 11
Documentary
Comedy—Musical—Romance
Drama—Mystery—Thriller
Comedy—Drama—Sci-Fi
Horror
Comedy—Romance—Thriller
Drama
Drama
Comedy
Children—Comedy

Column number of V: 15
Drama—Thriller
Drama
Drama—Romance
Action—Adventure—Drama—Thriller
Action—Crime—Thriller
Comedy—Horror
Drama
Documentary
Drama—Horror—Thriller
Drama—War

Column number of V: 19
Crime—Drama
Horror—Sci-Fi
Comedy—Horror
Horror
Action—Comedy
Comedy—Drama
Drama—Horror—Thriller
Drama—Romance
Comedy—Drama—Sci-Fi
Action—Drama—Sci-Fi

From the genre list, it can be seen that the top 10 movies belong to a small collection of genres. Each latent factor tends to group movies from small particular groups of genres, e.g. latent factor 15 seems to have most movies from the action and drama group, latent factor 7 seems to have movies from the action, drama genres etc.

10 Question 10

1. In this question, we are asked to design a Matrix factorization with bias (MF-bias) CF to predict the ratings of the movies in the MovieLens dataset and explore the relationship between the prediction error and the number of latent factors. This is similar to NNMF CF optimization program but with bias terms on each user and item as follows:

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

The user bias term models user's traits to give polar ratings compared to the average, while the item bias term models how a certain item is rated compared to the average ratings. Integrating the bias information helps better model such user or movie-specific noise or outliers in the generated embeddings. The optimization program can be thought of as a singular value decomposition (MF) problem. The prediction function is given by:

$$\hat{r}_{ij} = \sum_{s=1}^{k} (u_{is} v_{js}) + b_i + b_j + \mu$$

μ is the mean of all ratings. The Surprise-Scikit library's MF() is used for this question to implement the MF CF on the movie rating information. Figure below shows the average prediction root RMSE and MAE for various number of latent factors.

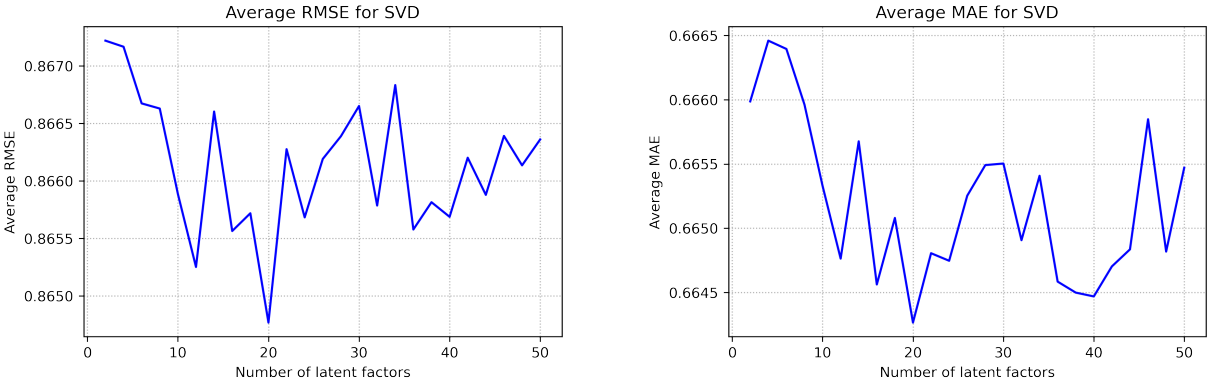

Figure 18: (Left) Average RMSE vs. k for MF CF (Right) Average MAE vs. k for MF CF

The above figure shows that the prediction error of MF CF is very consistent within a very tiny range of error across all latent factors (do not get confused by the erratic shapes of the plots, the actual error is varying within a very tiny range if you notice the y-axis). In fact, MF performs better than both k-NN and NNMF CF. This is expected because of several reasons:

- MF produces a hierarchical and geometric basis ordered by relevance, producing embeddings with the most relevant features higher in the hierarchy. Therefore, embeddings produced by MF with higher values of k is not adding any significant distinguishable information, neither subtracting any important semantic information (due to noise) thanks to the ordering of the features.
- As there are no constraints on U and V, MF is able to better represent the higher-dimensional feature matrix, providing a deep factorization with low information loss in both high and low dimensions.
- The embeddings produced by MF are deterministic and unique.
- MF takes into account user and movie-specific bias information and normalizes them appropriately to reduce sensitivity to outliers and noise.

2. In this section the optimum value of the number of latent factors for which the RMSE and MAE errors are minimal is found for MF CF (MF with bias). From the above figure, we see that this occurs at k = 20 for RMSE, with the minimum average RMSE = 0.864768 and k = 20 for MAE, with the minimum average MAE = 0.664265.Hence we take k = 20, which is also closer to the actual number of movie genres (19).

3. Here, we design a MF with bias with predictions being made on the popular movie trimmed test set. Figure 16 shows the average prediction RMSE for various number of latent factors.

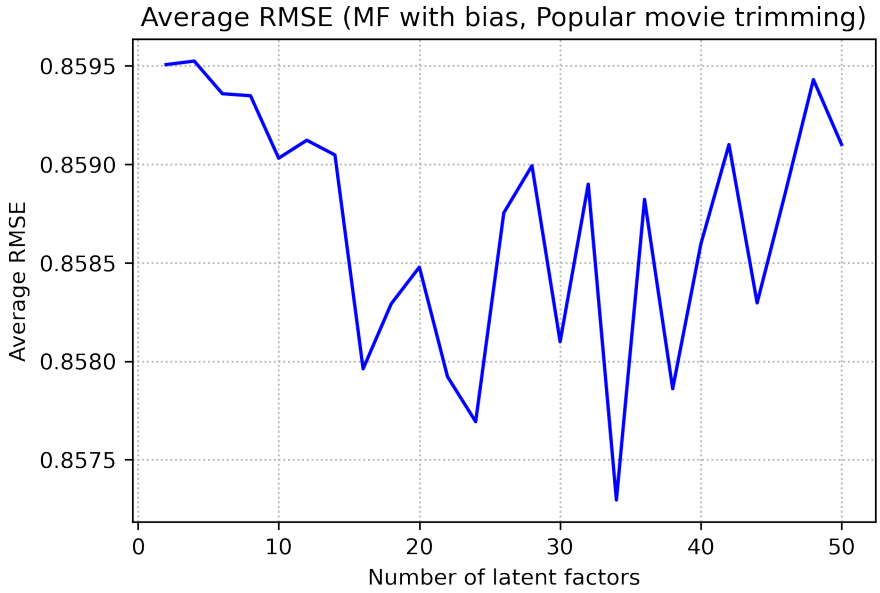

Figure 19: Average RMSE vs k for MF CF on popular movie trimmed test set

The minimum average RMSE for MF CF on the popular movie trimmed test set is 0.857295. We have provided an explanation of the excellent consistency of MF prediction error across all the latent factors.

4. In this section, we design a MF with bias) with predictions being made on the unpopular movie trimmed test set. The following figure shows the average prediction RMSE for various number of latent factors

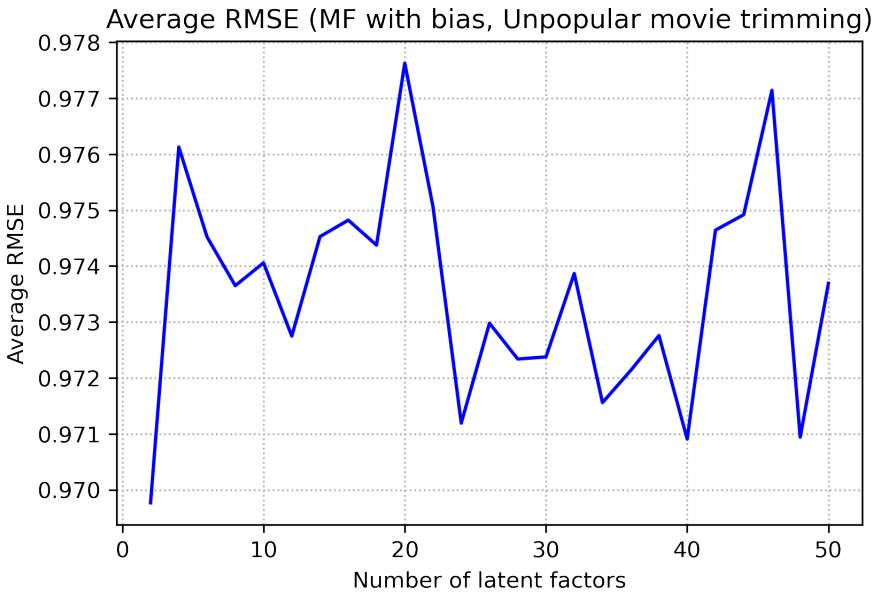

Figure 20: Average RMSE vs k for MF CF on unpopular movie trimmed test set

The minimum average RMSE for MF CF on the unpopular movie trimmed test set is 0.969766. This is larger than the minimum average RMSE on the popular movie trimmed test set

- The error is consistent across all the latent factors, which is explained in previous question.
- In addition, the error is the lowest for unpopular movie trimmed set compared to NNMF or k-NN user-based CF. This is because of MF's ability to model the bias for rarely rated items thanks to bias terms in the loss function, which is absent in k-NN or NNMF CF. As a result, MF generates lower prediction errors when making inferences on outliers.
- The error is still however, larger than the popular movie trimmed set for MF CF. The explanation which we provided for NNMF (Question 20) also holds here to explain why this happens

5. In this subquestion, we design a MF CF (MF with bias) with predictions being made on the high-variance movie trimmed test set. Figure below shows the average prediction RMSE for various number of latent factors.

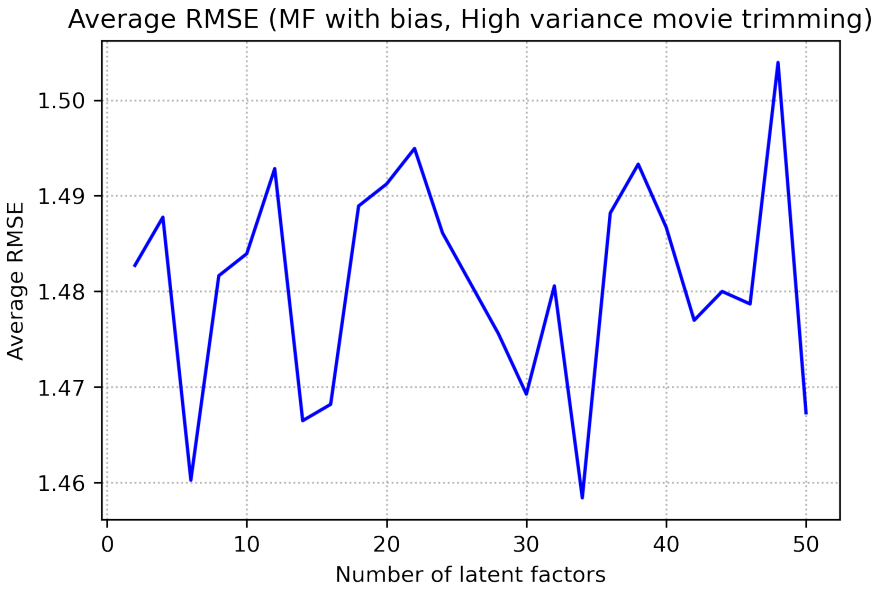

Figure 21: Average RMSE vs k for MF CF on high-variance movie trimmed test set

The minimum average RMSE for MF CF on the high-variance movie trimmed test set is 1.458406. As expected, this is much larger than the prediction error made by MF-CF on popular and unpopular movie trimmed set. We explained why this happens for NNMF in previous question, and the same explanation holds.

6. For this section, we plot the ROC curves for MF with bias for best value of k found in previous section, which was 20, for thresholds [2.5, 3, 3.5, 4]. The figure below shows the ROC curves for the four thresholds for MF CF with 20 latent factors.

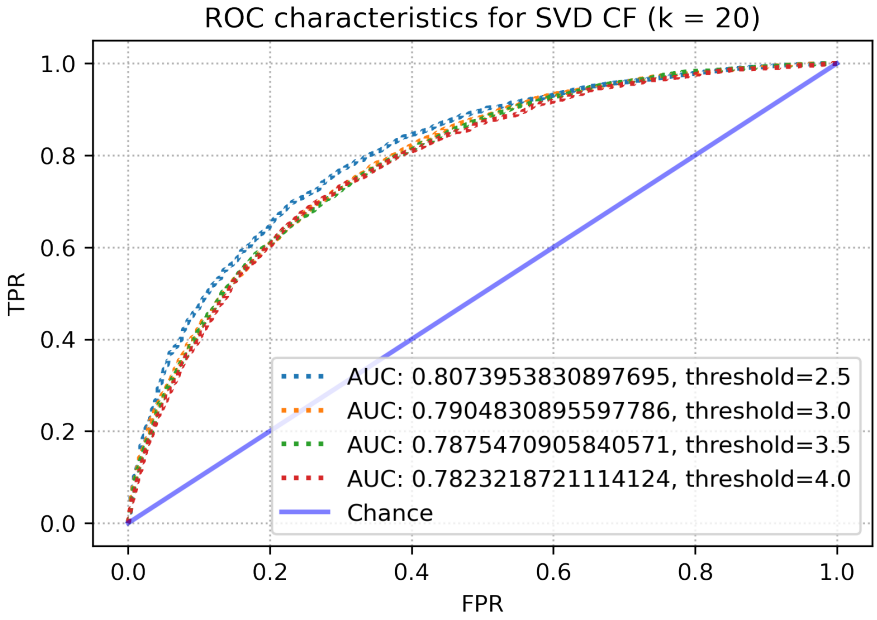

Figure 22: ROC curves for various thresholds for MF CF with 22 latent factors

The AUC for each threshold are as follows:

Threshold: 2.5, AUC: 0.807395
Threshold: 3.0, AUC: 0.790483
Threshold: 3.5, AUC: 0.787547
Threshold: 4.0, AUC: 0.782321

It can be observed that as the threshold increases, the AUC decreases monotonically for MF CF until threshold = 4.0, at which point it starts increasing.

11 Question 11

Designing a Naive Collaborative Filter

In this question, we are asked to design a naive collaborative filter to predict the ratings of the movies in the Synthetic Movie Lens dataset. The naive CF simply returns the mean ratings of a user on past items as the rating for the new item. The process for a naive collaborative filter involves 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).

$r_{ij} = \mu_i$

Then compute the RMSE for this fold and repeat the procedure for all the 10 folds.

- **Entire testset** - The average RMSE is computed by averaging the RMSE across all the 10 folds. It is just a one-shot prediction on the test folds based on the mean ratings in the entire dataset. The average RMSE for the entire testset for 10 test folds was **0.9347**.
- **Popular movie trimmed set** - In this question, we are asked to design a naive collaborative filtering with predictions being made on the popular movie trimmed test set. The average RMSE for 10 test folds for naive CF is **0.9323**. This is slightly lower than the RMSE for entire testset, as the popular movie trimmed set doesn't have any outliers present.
- **Unpopular movie trimmed set** - In this question, we are asked to design a naive CF with predictions being made on the unpopular movie trimmed test set. The average RMSE for 10 test folds for naive CF was **0.9713** in this case. This is slightly higher than the RMSE obtained on popular movie trimmed test set and entire test set. This is because the unpopular movie trimmed set will have some outliers present.
- **High Variance movie trimmed set** - In this question, we are asked to design a naive CF with predictions being made on the high-variance movie trimmed test set. The average RMSE for 10 test folds for naive CF was **1.453** in this case. This is much higher than the RMSE obtained on popular and unpopular movie trimmed test set. This is because the high variance trimmed set has the maximum number of outliers present with very low generalization.

12 Question 12

In this question, we are asked to plot the ROC curves for MF with bias, k-NN user-based CF and NNMF CF for thresholds of 3. We used 20 neighbors for k-NN user-based CF, 20 latent factors for MF with bias and 18 latent factors for NNMF, best results found in previous questions. The following shows the ROC curves for all three CF

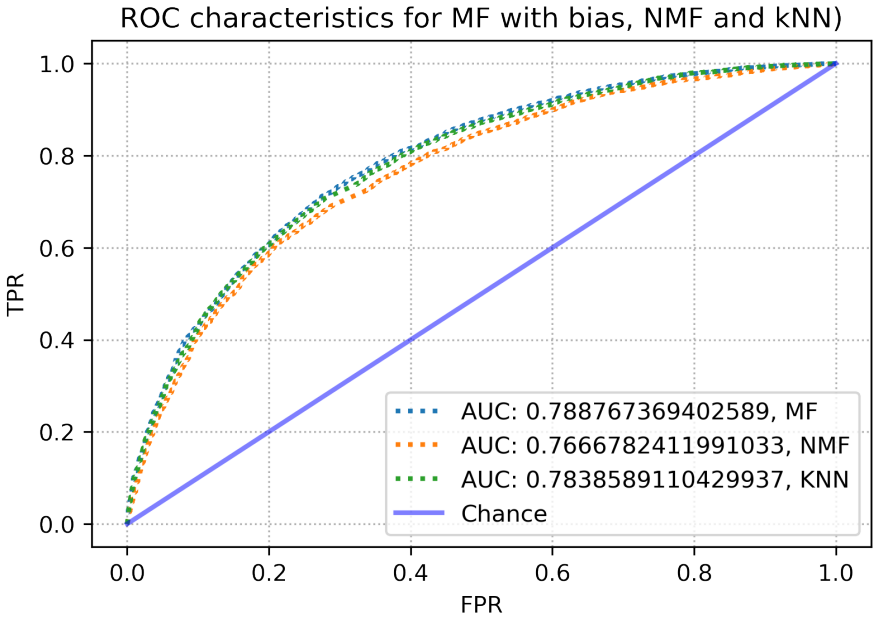

Figure 23: : ROC curves for k-NN user-based CF (k = 20), NNMF CF (k = 18) and MF CF (k = 20)

The AUC for each CF are as follows:

k-NN CF - 0.7838
NNMF CF - 0.7666
MF CF - 0.7887

From the above figure, it can be inferred that MF CF performs best among all the CF, followed by k-NN CF and NNMF-CF coming last. We explain the performance as follows:

MF with bias vs NMF

- MF with bias produces a hierarchical and geometric basis ordered by relevance, producing embeddings with the most relevant features and traits in the ratings matrix higher in the hierarchy. Thus, embeddings produced by MF are robust to outliers and noise in the ratings thanks to the ordering of the features. NMF, on the other hand, does not consider the geometry in the ratings matrix
- MF with bias takes into account user and movie-specific bias information and normalizes them appropriately to reduce sensitivity to outliers and noise.
- The embeddings produced by MF with bias are unique and deterministic, whereas NMF is non-unique and stochastic, with no guarantees of convergence to the optimal U and V each time the function is called.
- MF with bias is able to better represent the higher-dimensional feature matrix due to no constraints on U and V, providing a deep factorization with low information loss. NMF on the

other hand, restricts U and V to be positive and has fewer optimal choices of elements in U and V compared to MF with bias

Why k-NN performs slightly worse than MF-CF:

- k-NN is not modeling the bias information separately for each user or item. As a result, it is more sensitive to outliers and rarely rated items
- k-NN is much less generalizable compared to latent-factor based models, as it cannot find semantic information and connections within the user-item ratings matrix while being sensitive to rarely rated items.
- k-NN performs inference directly on the sparse ratings matrix, which yields poor prediction accuracy in high-dimensional space. This also hurts the scalability of the recommender system. High-dimensional inference requires large amounts of training data to work properly, which is absent as the ratings matrix is sparse.

13 Question 13

Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. In other words, precision is a measure of the exactness or reliability of a CF. Low precision indicates high number of false positives. In terms of recommendation systems, precision indicates the percentage of items the user actually liked out of the set of recommended items.

Recall, on the other hand, is a measure of sensitivity or completeness (information retrieval capacity) of a CF. So recall is the measure of our model correctly identifying True Positives. Low recall indicates high number of false negatives. In terms of recommendation systems, recall indicates whether all the items the user likes were recommended to the user or not.

14 Question 14

1. In this part of the question, we are asked to 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 previous question, 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. The steps undertaken to obtain the curves are as follows:

   - Find the list of movies G liked by the user. The set only contains those movies that have a score greater than 3 as the threshold is given as 3.
   - Create a dictionary of all movies rated by each user.
   - Split the dataset into 10 pairs of training set and test set as we have to perform 10 fold cross validation.
   - Drop those users in the test set who either rated less than t items (in the dictionary) or have no items in the set G.
   - Compute predicted ratings with the chosen CF (in this case k-NN with Pearson similarity metric) for each user
   - Sort the predicted ratings in descending order and select the first t items to yield S(t).
   - Calculate precision and recall for each user using the formula provided in the question
   - Take the mean of the precision and recall across as users and across all test folds

   The following figure shows all three curves for k-NN user-based CF for t ranging from 1 to 25.

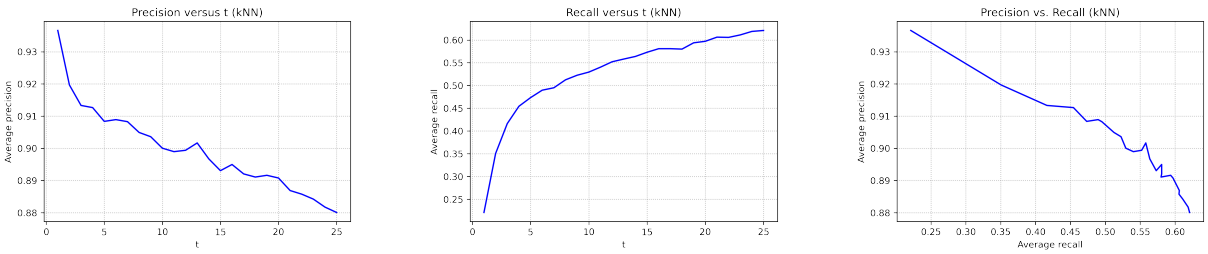

Figure 24: Precision vs t, Recall vs t and Precision vs Recall curves for k-NN user-based CF with 20 neighbors

From the above figure, we can make the following comments about the curves:

- As t increases, the average precision decreases. This is expected because the CF is more likely to generate false positives as the number of items recommended to the user increases. In other words, as S(t) increases, the value of S(t) $\cap$ G is likely to drop as the CF is likely to suggest something that the user does not like. However, the reduction in precision is only around 5 percent for a 25X increase in t for k-NN, so we can say that the precisions are overall consistent across t.
- As t increases, the average recall increases. This is because with a higher value of t, the probability that the predicted ratings will encompass all of the liked movies of the user (completeness) increases as well. The range of recall is much wider than that of the precision, with a 35 percent gap between the lowest recall and highest recall for a 25X increase in t for k-NN. This indicates that recall is more sensitive to how many items are being suggested compared to precision.
- As average recall increases, the average precision decreases. With a higher recall, apart from having a higher probability of including true positives in the recommendation list, the CF is also more likely to include items that are not liked by the user, which leads to a lower precision. In other words,a higher recall leads to a lower precision and vice versa.

2. In this question, we are asked to show the plots of precision vs. t, recall vs. t and precision vs.accuracy curves for NNMF CF with the best value of the number of latent factors found previous question, which is 18. The steps undertaken to obtain the curves are similar to the steps listed in previous question. Figure below shows all three curves for NNMF CF for t ranging from 1 to 25.

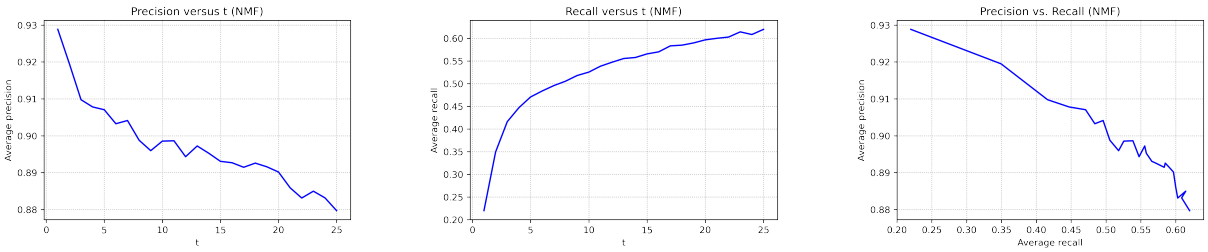

Figure 25: Precision vs t, Recall vs t and Precision vs Recall curves for k-NN user-based CF with 20 neighbors

From the figure, we can make the following comments about the curves:

- As t increases, the average precision decreases. This is expected because the CF is more likely to generate false positives as the number of items recommended to the user increases. In other words, as S(t) increases, the value of S(t) $\cap$ G is likely to drop as the CF is likely to suggest something that the user does not like. However, the reduction in precision is only around 4 percent for a 25X increase in t for NNMF, so we can say that the precision is overall consistent across t. This range is lower than what was obtained for k-NN.
- As t increases, the average recall increases. This is because with a higher value of t, the probability that the predicted ratings will encompass all of the liked movies of the user (completeness) increases as well.The range of recall is much wider than that of the precision, with a 35 percent gap between the lowest recall and highest recall for a 25X increase in t for NNMF. This indicates that recall is more sensitive to how many items are being suggested compared to precision.
- As average recall increases, the average precision decreases. With a higher recall, apart from having a higher probability of including true positives in the recommendation list, the CF is also more likely to include items that are not liked by the user, which leads to a lower precision. In other words, a higher recall leads to a lower precision and vice versa.

3. In this subsection, we show the plots of precision vs. t, recall vs. t and precision vs.accuracy curves for MF with bias with the best value of the number of latent factors found in previous question, which is 20. The steps undertaken to obtain the curves are similar to the steps listed in previous question. The following question shows all three curves for MF CF for t ranging from 1 to 25.

   From the above figure we come to the following conclusion:

- As t increases, the average precision decreases. This is expected because the CF is more likely to generate false positives as the number of items recommended to the user increases. In other words, as S(t) increases, the value of S(t) $\cap$ G is likely to drop as the CF is likely to suggest something that the user does not like. However, the reduction in precision is only around 5 percent fora increase in t for MF, so we can say that the precisions are overall consistent across t.

- As t increases, the average recall increases. This is because with a higher value of t, the probability that the predicted ratings will encompass all of the liked movies of the user (completeness) increases as well. The range of recall is much wider than that of the precision, with a 35 percent gap between the lowest recall and highest recall for a 25X increase in t for MF. This indicates that recall is more sensitive to how many items are being suggested compared to precision
- As average recall increases, the average precision decreases. With a higher recall, apart from having a higher probability of including true positives in the recommendation list, the CF is also more likely to include items that are not liked by the user, which leads to a lower precision. In other words,a higher recall leads to a lower precision and vice versa.

4. In this question, we are asked to plot the precision vs. recall curves for k-NN user-based CF, NNMF CF and MF CF (MF with bias) obtained in Questions 36, 37 and 38 on the same plotand compare their performance. The following figure shows the plot.

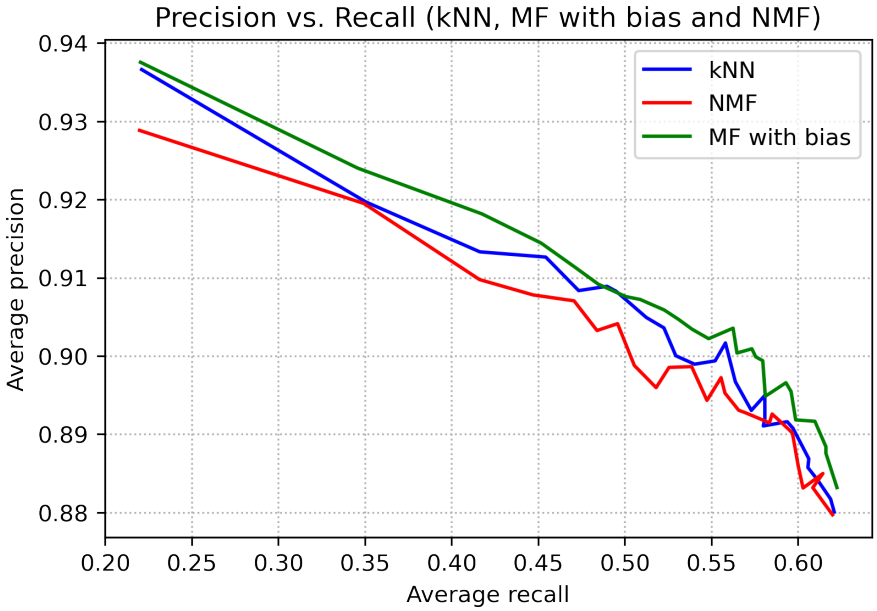

Figure 26: Precision vs Recall curves for k-NN, NNMF and MF CF

From the figure, it can be observed that MF CF (MF with bias) provides the best performance as its precision drops slower with increase in recall compared to NNMF or k-NN CF while also maintaining a higher precision for each recall value compared to the latter two. k-NN user-based CF provides the 2nd best performance, followed by NMF-CF. In other words, we can say that MF CF provides the most relevant recommended list (set of items most likely to be favored by the user) to each user, followed by k-NN CF and lastly NNMF-CF.