**Script:**

\*\* have a look at the other data files given

* **Union and Missing Values**

This solution first computes the union of items rated by either user ‘a’ or ‘u’, rather than taking the intersect. This allows for a greater number of samples in the training data. As we are finding the union and it is not confirmed that both users have rated that item, it is likely that there will be missing values. These missing values are replaced with the user’s mean rating for that particular category. Calculating the union is helpful as the number of movies a user may have rated could be extremely small and finding movies that both users rated could be even smaller. By calculating the union and including movies both users have rated allows the sample size to be increased. This allows for a potentially more accurate model, therefore improving the performance of the user KNN-based Collaborative Filtering method. The missing values being replaced with the user’s mean rating for movies in that category is necessary. This is because if the missing values are not replaced, then a rating of 0 would be given. This would suggest to the model that the user did not like the movie, however, a user not watching or rating the movie does not mean that the user doesn’t like the movie. Using the mean for that particular category rather than the overall mean allows the model to differentiate between the user’s interests. For example, a higher mean rating for action movies compared to comedy movies tells the model to find other users who also like action movies and recommend those types of movies. This is better than just using the overall mean as done in the original method which treats all movies in the database as the same which could lead to incorrect predictions. For this reason, finding the union and imputing missing values with the mean rating for movies in that category allows for better performance

* **Weight Considering Item’s Popularity**

When doing the similarity calculation, the solution provided in the report considers the popularity of the different items unlike the original method. Considering the weight of each item is important as it allows the method to differentiate between popular and less popular items, treating them differently. This is important as unlike the original model, this considers the impact of the long-tail theory. The long-tail theory suggests that there is a small selection of items with extremely high popularity and demand, and most items are in the long tail and are not that popular. The solution in the report does this by squaring the weight of the popularity in the formula to find similarity. The weight is calculated by finding the log of the total number of users in the database divided the total number of users that rated that item. Here, a larger number of users that rated the item would result in a smaller value for the weight, and therefore hold less significance. Items that are less popular and have a smaller number of users that rated the items would have a higher weight and therefore have greater significance. This means that the solution provided in the report is less likely to just predict super popular movies and is more likely to predict movies in the long tail, and therefore less popular movies. This allows for a wider range of recommendations to be given. This consideration of the weight ultimately improves the performance of the user KNN Collaborative Filtering method.

* **Code**

First the value for ‘P’ was calculated. This was done by finding the sum of all non-zero values in each column. Non-zero values suggest that this item was rated by the user.

Inside the nested ‘for’ loop a mask was created for user ‘a’ and user ‘u’. The mask contained a value of true if the rating was greater than 0 and had therefore rated that item, and false if the rating was zero, suggesting the user had not rated that item.

Then the union set was found. This was by finding the union between the masks created for this user. This contained the index of an item where either user ‘a’ or user ‘u’ had rated the item, and therefore had ‘true’ in the mask.

Then, for users ‘a’ and ‘u’, for any items they have not rated in the union is replaced with the user’s mean rating. This is done by first calculating the mean by dividing the sum of all of the user’s ratings by the total number of items the user had rated. This is done through the use of np.clip which has a maximum value of 1. This means that any rating greater than 1, becomes 1. This is done so that each item rating was counted once. Then a ‘for’ loop is used to go through each index in the union, and if that index in the user’s vector is empty and had a rating of 0, this is replaced with the user’s mean. New vectors are then created with only items in the union set.

Then the weight of each items popularity is computed. This is done by finding the log of the total number of users divided by the value of P which was calculated earlier. Then, the numerator of similarity formula is calculated. This is done by calculating the sum of the square of their weight multiplied by user ‘a’ rating for the item minus the user ‘a’ mean rating multiplied by the same thing for user ‘u’. This sum is done for each item in the union set. Then, the denominator section for user ‘a’ is calculated. This is the square root of the weight squared multiplied by the square of the different between user ‘a’ rating for that item minus user ‘a’ mean rating for the item. The sum is once again calculated for each item in the union set. The same is done for the denominator section for user ‘u’. The overall denominator is found my multiplying the two sections. The similarity is then found by dividing the numerator by the denominator. This similarity is then added to the user correlation matrix.

A matrix is then created to store the predictions. The K Nearest Neighbours selection is then done using k=20. Then similar users are found based on the current user using the k-value. This is then sorted in descending order. Argsort is used as it allows us sort in a way which returns the index of each user once it has been sorted. Then the user itself is removed as they will obviously be the most similar to themself.

Then, the similar items are used to calculate the coefficient values. The mean rating for the current user is then calculated. First, all the similar users are obtained from the training dataset. Then a mask is created based on if the user has rated that item or not. The Boolean values are then converted into floats where ‘1’ is for True and ‘0’ is for False. This is done so we can count all the time we get True, and therefore get the number of items the user has rated. The mean rating is then calculated by the getting the sum of each item the user has rated and dividing it by the total number of items the user has rated. We then look at the mean rating of similar users. Similarly, a mask is created to see if similar user had rated each item or not. We again convert Booleans to float and calculate the sum to get the total number of items rated by similar users.

The sum of the ratings for similar users in then found. The mean rating for similar users is found by dividing the total sum of the ratings by the number of items rated by that user. A mask is then created for user ‘u’ based on whether the item has been rated or not. The numerator of the formula before the summation is then calculated. Then, the final prediction is calculated by substituting all values into the formula. The prediction rating calculated is then added to the matrix. Np.clip is used to do this to ensure the prediction is between 0 and 5, so any predictions less than 0 become 0 and predictions greater than 5 become 5.

Finally, the evaluate function is used to get the MAE, and RMSE values. Using the code we get a MAE value of around 0.779 and a RMSE value of 0.9757

**– Explain how the solution in the provided report can improve the performance**

**of the user KNN-based Collaborative Filtering method by using your own**

**language clearly and completely.**

Union and filling in the missing values.

The formula.

This solution first computes the union of items rated by either user ‘a’ or ‘u’, rather than taking the intersect. This allows for a greater number of samples in the training data. As we are finding the union and it is not confirmed that both users have predicted that item, any missing values are replaced using the user’s mean rating. When doing the similarity calculation, the solution provided in the report considers the popularity of the different items unlike the original method. This is done by calculating the log of the total number of users in the database and dividing it by the number of users that rated that item.

**– Explain why the solution in the provided report can improve the performance**

**of the user KNN-based Collaborative Filtering method by using your own**

**language clearly and completely.**

Union and filling in the missing values. (Why?)

The formula (Why?)

Calculating the union is helpful as the number of movies a user may have rated could be extremely small. By calculating the union and including movies both users have rated allows the sample size to be increased. This allows for a potentially more accurate/precise model, therefore improving the performance of the user KNN-based Collaborative Filtering method. The missing values being replaced with the user’s mean rating also improves the rating. This is because if the missing values are not replaced, then a rating of 0 would be given. This would suggest to the model that the user did not like the movie, however, a user not watching the movie does not mean that the user doesn’t like the movie. Therefore, by imputing the mean, this problem does get solved. Considering the weight of each item is important as it allows the method to differentiate between popular and less popular items. This means that more popular items have less significance compared to less popular items. This considers the long-tail effect which the original model does not. So, instead of just recommending popular movies, these movies have a lower significance. This models allows for the movies to be recommended in the long-tail which is made up of less popular movies, and these movies have a higher significance in the model. This is because when dividing the total number of users by the number of users who have rated the movie, this number is higher with a lower value for P.

**– Explain how you implement the solution clearly and completely.**

Explain the code.

ComRV is not found as it does not have anything to do with the MAE and RMSE.

Maybe, in each slide have the how and the why for that one part on the same slide.