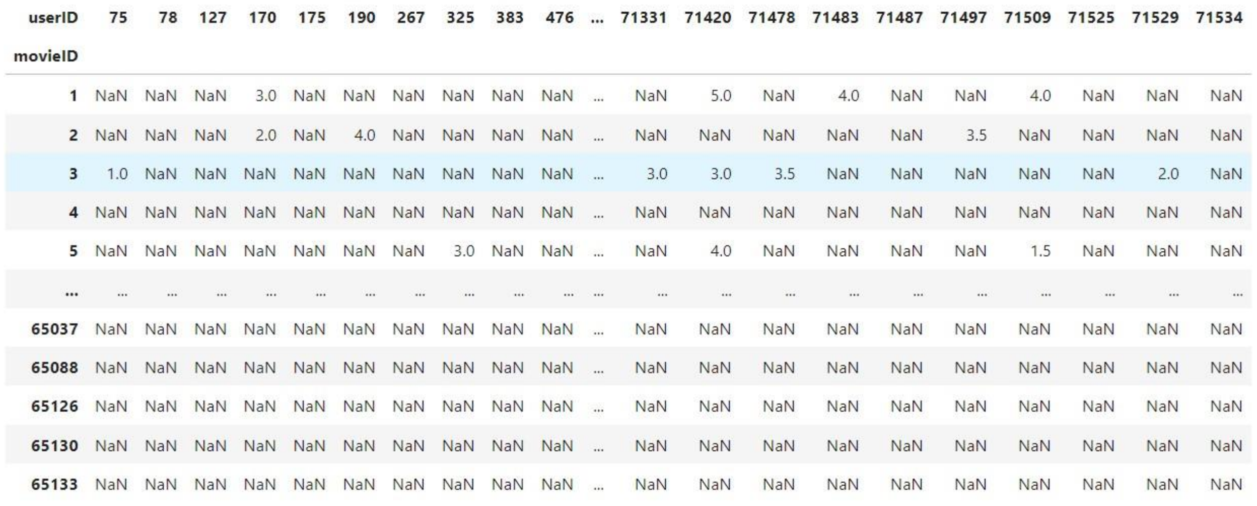
# Recommendation System Project Report

**RMSE (average RMSE over multiple iterations of 1000 datapoints sampled randomly):** 0.85

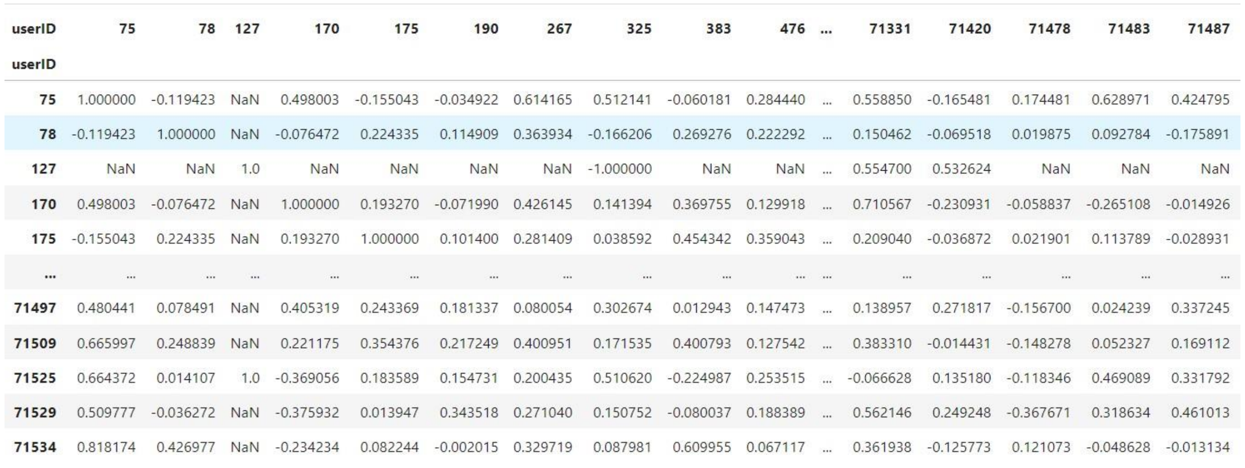
For the project:

We started off by combining all the data to form item profiles for the movies. However, tags and user_taggedmovies were not included in the combined data. Following this the item profiles were merged with the training data with respect to the movieIDs.  This data was then converted into a utility matrix, between userIDs and movieIDs, the overview of utility matrix is as shown below:



| userID | 75 | 78 | 127 | 170 | 175 | 190 | 267 | 325 | 383 | 476 | ... | 71331 | 71420 | 71478 | 71483 | 71487 | 71497 | 71509 | 71525 | 71529 | 71534 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| movieID | | | | | | | | | | | | | | | | | | | | | |
| 1 | NaN | NaN | NaN | 3.0 | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 5.0 | NaN | 4.0 | NaN | NaN | 4.0 | NaN | NaN | NaN |
| 2 | NaN | NaN | NaN | 2.0 | NaN | 4.0 | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | 3.5 | NaN | NaN | NaN | NaN |
| 3 | 1.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | 3.0 | 3.0 | 3.5 | NaN | NaN | NaN | NaN | NaN | 2.0 | NaN |
| 4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 5 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 3.0 | NaN | NaN | ... | NaN | 4.0 | NaN | NaN | NaN | NaN | 1.5 | NaN | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 65037 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 65088 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 65126 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 65130 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 65133 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

Following this we made a user correlation matrix to display relativity among different users. The overview of the correlation matrix is as follows:



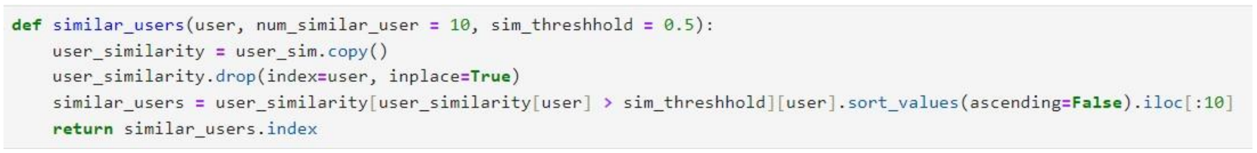
| userID | 75 | 78 | 127 | 170 | 175 | 190 | 267 | 325 | 383 | 476 | ... | 71331 | 71420 | 71478 | 71483 | 71487 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| userID | | | | | | | | | | | | | | | | |
| 75 | 1.000000 | -0.119423 | NaN | 0.498003 | -0.155043 | -0.034922 | 0.614165 | 0.512141 | -0.060181 | 0.284440 | ... | 0.558850 | -0.165481 | 0.174481 | 0.628971 | 0.424795 |
| 78 | -0.119423 | 1.000000 | NaN | -0.076472 | 0.224335 | 0.114909 | 0.363934 | -0.166206 | 0.269276 | 0.222292 | ... | 0.150462 | -0.069518 | 0.019875 | 0.092784 | -0.175891 |
| 127 | NaN | NaN | 1.0 | NaN | NaN | NaN | NaN | -1.000000 | NaN | NaN | ... | 0.554700 | 0.532624 | NaN | NaN | NaN |
| 170 | 0.498003 | -0.076472 | NaN | 1.000000 | 0.193270 | -0.071990 | 0.426145 | 0.141394 | 0.369755 | 0.129918 | ... | 0.710567 | -0.230931 | -0.058837 | -0.265108 | -0.014926 |
| 175 | -0.155043 | 0.224335 | NaN | 0.193270 | 1.000000 | 0.101400 | 0.281409 | 0.038592 | 0.454342 | 0.359043 | ... | 0.209040 | -0.036872 | 0.021901 | 0.113789 | -0.028931 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 71497 | 0.480441 | 0.078491 | NaN | 0.405319 | 0.243369 | 0.181337 | 0.080054 | 0.302674 | 0.012943 | 0.147473 | ... | 0.138957 | 0.271817 | -0.156700 | 0.024239 | 0.337245 |
| 71509 | 0.665997 | 0.248839 | NaN | 0.221175 | 0.354376 | 0.217249 | 0.400951 | 0.171535 | 0.400793 | 0.127542 | ... | 0.383310 | -0.014431 | -0.148278 | 0.052327 | 0.169112 |
| 71525 | 0.664372 | 0.014107 | 1.0 | -0.369056 | 0.183589 | 0.154731 | 0.200435 | 0.510620 | -0.224987 | 0.253515 | ... | -0.066628 | 0.135180 | -0.118346 | 0.469089 | 0.331792 |
| 71529 | 0.509777 | -0.036272 | NaN | -0.375932 | 0.013947 | 0.343518 | 0.271040 | 0.150752 | -0.080037 | 0.188389 | ... | 0.562146 | 0.249248 | -0.367671 | 0.318634 | 0.461013 |
| 71534 | 0.818174 | 0.426977 | NaN | -0.234234 | 0.082244 | -0.002015 | 0.329719 | 0.087981 | 0.609955 | 0.067117 | ... | 0.361938 | -0.125773 | 0.121073 | -0.048628 | -0.013134 |

For this process we made use of person correlation, which depends on covariance between two variables. Pearson correlation is a measure that quantifies the strength and direction of the linear relationship between two continuous variables. The reasons behind the use of 'Pearson correlation' over cosine similarity are as follows:

- Simplicity in implementation.
- Accepts data with missing values as opposed to cosine similarity, which does not.
- Pearson correlation can detect both positive and negative relationships, while cosine similarity only measures similarity and cannot detect if the relationship is negative.
- Pearson correlation adjusts for the fact that different users may have different rating scales. For example, one user might tend to give high ratings, while another might rate more conservatively. Pearson correlation considers these differences in rating behavior by standardizing the ratings (subtracting the mean and dividing by the standard deviation).

However, due to this implementation we faced issues including higher time complexity and negative impacts on accuracy as the Pearson correlation does not account for the features. This was unavoidable due to time constraints.

Following this, we defined a function that fetches a defined number of users similar to the user in focus using the users correlation matrix. The similar users are picked with respect to a similarity threshold. The implementation is as follows:



```python
def similar_users(user, num_similar_user = 10, sim_threshhold = 0.5):
    user_similarity = user_sim.copy()
    user_similarity.drop(index=user, inplace=True)
    similar_users = user_similarity[user_similarity[user] > sim_threshhold][user].sort_values(ascending=False).iloc[:10]
    return similar_users.index
```

As for the implementation of the prediction model, we took an unusual approach.

For case 1, we based our prediction on similar user ratings and had ratings for the movie in focus by similar users. To determine similar users, we made use of the 'similar_users' function defined previously. We then considered the ratings of the movie by similar users and computed the mean of these ratings to predict the rating for the movie by the respective user.

Coming to case2: we lacked enough ratings for the movie by similar users.  In order to deal with this issue, we considered ratings of the movie given by all users present in the data frame. Considering all the ratings, we computed the mean of these ratings to determine the predicted rating for the movie.

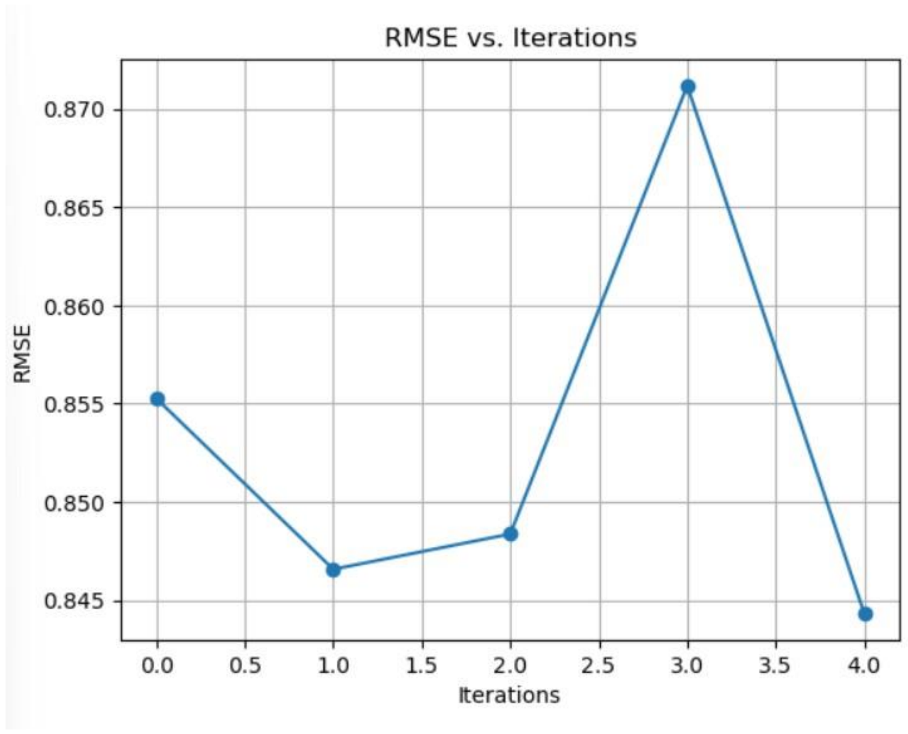
Finally, we had to deal with a subcase in case 2, where there was no data about the movie in focus. For this case, we considered ratings of all movies across the dataframe. We then computed the mean of all these ratings for the predicted rating.

## Model Evaluation:

For model evaluation we used root mean squared error. Due to time complexity issues, we ran 5 (default) iterations of randomly sampled datapoints (1000) to compute the RMSE for each iteration. The computation of the RMSE was done as follows:

- Mean squared error was computed using the function ‘mean_squared_error’ from the ‘metrics’ module of the ‘sklearn’ (scikit-learn) library.
- RMSE was then computed by calculating the square root of the mean squared error.

the distribution of RMSE over the iterations is as follows:



The average RMSE over these iterations was ‘0.8531445057806735’.

Unfortunately, we were also unable to perform any hyperparametric tuning due to time constraints.