CS333 HW5

Jeffrey Luo (jl834) Nathan Kim (njk24)

1.
    a. Movie ID: 318, Average Rating: 4.49
    b. Movie ID: 231, Average Rating: 2.64
    c. 74.28 average number of movies rated
2.
    a. To make the calculation of angular distance between two users efficient, a map was created where the keys were user_ids. Each user_id was paired with another map. This nested map had keys equal to movie_ids that the user_id had rated, and the values of this nested map were the ratings that the user_id had given for the movie_id.
    To calculate the angular distance between two users, two O(1) operations were done to get the nested maps for the two user_ids. Then, a loop was run over all movie_ids in a user_id's nested map (O(n) operation). For each iteration of this loop, we check if the other user_id's nested map contains the movie_id that we are currently inspecting (O(1) operation). If it does, we perform the necessary mathematical operations, and increment a counter representing the number of movies that both users have rated. After the loop is done, if the counter is greater than or equal to the threshold, return the mathematical formula. Otherwise, return 1.
    The overall time complexity of this algorithm is O(n), where n is the number of movies rated by a user.

3.
    a. See code.
    b. Recommendations for user 2:
        1. Close Shave, A (1995) - 4.7
        2. Chinatown (1974) - 4.7
        3. Maltese Falcon, The (1941) - 4.7
        4. Pulp Fiction (1994) - 4.653846153846154
        5. Usual Suspects, The (1995) - 4.642857142857143
    c. My Recommendations:
        1. Close Shave, A (1995) - 4.7
        2. Casablanca (1942) - 4.65
        3. Killing Fields, The (1984) - 4.625
        4. Shallow Grave (1994) - 4.625
        5. Cinema Paradiso (1988) - 4.625

    These do seem like movies I would be interested in watching. I was a little shocked that no Batman movies appeared, since I rated a few Batman movies highly (making sure not to rate every single Batman movie).
    d. Step i) (finding the k-nearest neighbors) takes the longest, with a time complexity of $O(\overline{nm} + n\,log(n))$. Step ii) (finding the set of unseen movies)

has a time complexity of $O(n\overline{m} + \overline{m})$. This is because the implementation requires looping over all neighbors, as well as every movie a neighbor has watched in order to add the movie to a set. This is $O(n\overline{m})$. After this loop, we need to remove any movies that the user has watched from the overall large set. This operation is $O(\overline{m})$. This results in an overall average runtime of $O(n\overline{m} + \overline{m})$. Step iii) has a time complexity of $O(n)$, and Step iv) has a time complexity of $O(n\ log(n))$. Therefore, step i) has the largest time complexity, making it the bottleneck.

e. We smooth it to reduce the effect that an outlier could have on smaller datasets. For example, when there have not been many people that have rated a movie, their ratings have an extremely strong effect on the average rating of that movie. By smoothing, we reduce this effect, giving a little more reliability to the ratings.

4.

a. Time elapsed: 37.486922556 seconds
RMSE(predictions) = 1.0535680214786332
RMSE(baseline) = 1.1713240371477085

b. RMSE(predictions) = 1.139089167798633
RMSE(baseline) = 0.4128841992463321

c. Increasing the *k* value will result in smaller errors. By having more neighbors, there is a higher chance that a certain movie will be seen by some of these neighbors, thus resulting in the average rating that neighbors give the movie being closer to the average rating that all users (not just the neighbors) give the movie. Overall, this would reduce the RMSE.