Assignment 3
Daniel Crawford
CS 6375 - Machine Learnings

Algorithm:

The algorithm for our collaborative filtering is very expensive, and thus it demands many optimizations to give us results in a given timeframe. Once fully implemented, the algorithm does X things:

1. Finds the average ratings from user A: O(n)
2. Find all users B which voted on target movie I: O(n)
3. Find all users C which have common ratings with user A: O(n^2)
4. Compute Pearson Coefficient for each user C times their vote on movie I minus their average vote. O(|users C|^2)
5. Compute absolute sum of Pearson Coefficients found for each user C and find inverse Os(n)
6. Now return computation results. O(1)

That leaves our algorithm at O(n^2) in the worst case. With a data set of size 3 million, n^2 is very complex. By observation, we can see that we can best improve our algorithm by looking at step 3 and 4. For step 3, we can improve our times from O(n^2) to O(|User Movies| * |Users that voted on Movie I|) by using a memory tradeoff. To do this, we simply group each MovieID by UserID and then return a set for each list of MovieIDs a user has. Transform this into a dictionary for O(1) access time, and we can accomplish a better speed. Set intersection is O(n) time, so we can find common movies between each other user very quickly now. This small optimization still leaves our algorithm at O(n^2) worst case, but on average it will produce more efficient results.

Evaluation:

The results of this algorithm show that it has gained a decent understanding of what a user's preferences are. Here is a quick list of the results from our test set.

Test sample size	100478
Train sample size	3255352
Root Mean Square Error	0.9494
Mean Absolute Error	0.7492

Our results show that room for improvement would be preferred, but there are decent estimates being made. Our mean absolute error shows that our algorithm was usually 0.7492 stars off from a user's true preference. Thus, our algorithm has a general understanding of what a user wants. However, there are issues in this algorithm that can be causing our errors to be higher. One assumption that is made here is that a users average vote will be applied to a preference if it can't determine a user that correlates with the target user. One method to avoid these errors is to use default voting. This will increase complexity on the algorithm but improve the performance on the test set. Root mean square error(RMSE) on the other hand shows us similar to mean absolute error what our differences are. Our RMSE also gives how skewed our

residuals from our algorithm. From observation, it seems there is a skewed fit on the data, but the results are still satisfactory.