Assignment 3
Daniel Crawford
CS 6375 - Machine Learnings

Algorithm:

The collaborative filtering algorithm is very expensive, and thus it demands many optimizations to produce good computational performance. 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| * n)
5. Compute absolute sum of Pearson Coefficients found for each user C and find inverse O(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 will take far too long. 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(|Target user Movies| * |Users that voted on Movie I|) by using a memory tradeoff. To do this, we generate a set of MovieIDs that each UserID has voted on, then index this using a dictionary. Set intersection is O(n) time, so we can find common movies between each other user faster than O(n^2) on average. 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

First of all, one thing to consider here is that our results are on a scale of five (i.e. one out of 5 stars). These results show that our error is usually about a star off from the actual rating. Our mean absolute error shows that our algorithm was usually 0.7492 stars off from a user's true preference. To be general, this performance allows us to to decide whether a user will like a movie or not. Where it struggles is determining at what extent will a user like a movie, so it may underestimate or overestimate how much a user actually enjoys a movie. 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 user's average vote will be applied to a preference if it can't find any other users that our target user correlates with. An example of when this can happen if a user has only one vote on an obscure movie. One method to avoid these errors is to use default voting. With default voting, we can assume a default vote for other users, and compute a more likely rating for a target user instead of canceling values out to 0. This will increase complexity on the algorithm but improve the performance on the test set. Root mean square error(RMSE) shows us similar to mean absolute error what our differences are. Our RMSE also gives how skewed our residuals from our algorithm. Our RMSE confirms that our algorithm is at least able to accomplish whether a user will like or dislike a movie, because the points are on average 0.9494 stars off from the actual rating.