I started by looking into parsing the data from the .txt file into a pandas DataFrame, and how to organize the ratings. I decided to treat each movie rating as one observation, and parsed the movie id rows and assigned them to their respective movie ratings. So each row in the DataFrame had the movie id, user id, rating number, and date. I also parsed the date and categorized them by groups of months. January, February, and March were represented with 1, April, May, and June with 2, and so on with 3 and 4. This is to loosely represent the months as seasons.

I solved the challenge of dealing with the .txt file and parsing it into a usable format in a pandas DataFrame. However, this method of parsing also meant that for the challenge of dealing with missing data, I automatically ignored any possible missing rating data–if the rating did not exist in the data.txt file, then it was not given a spot in the DataFrame. This makes dealing with the missing ratings easier, as any missing ratings from users are not included, so finding a suitable method of imputation is not necessary. But it does come with the drawback that some information is lost. If a user did not rate a movie, it could mean they were not aware of the movie’s existence, knew of the movie but did not watch it, or they even could have already seen the movie, but chose not to rate it. Since I chose to drop missing values, none of this information is included in the data set–only the actual ratings that users gave.

I saved the parsed movie rating data into a DataFrame, and randomly sampled one rating from each unique movie id and saved it into the test set and put the rest into the training set in order to satisfy the test set requirements from the spec sheet.

Since the data set was large, with over 2.7 million observations even without imputed missing data, I only tried simpler models in order to train in a reasonable amount of time. I initially tried using the movie id and user id in my models to try to predict the rating, but could not get an RMSE below the 1.2s. I added the month the rating was made at first and eventually grouped the months as mentioned above after the months alone were not improving the models. I still could not get much improvement, but decided to drop the user id for the models as user ids aren’t predictive enough of the rating of a movie, and to only use the movie id and the month grouping.

I discarded any linear methods fairly quickly as the predictors were not linear in nature, especially not the movie ids or user ids. I tested out using neural networks, but it was difficult to map the function well and the RMSE did not really improve either. I thought the best model would be some sort of tree based model, as the data is mostly categorical–movie id, user id, and month grouping are numbers, but really represent categorical information. I tried out AdaBoosting with weak learner trees, but the Random Forest Regressor saw the greatest drop in RMSE. Interestingly enough, the n\_estimators parameter did not have a huge effect on the RMSE, so I used a lower amount to save time.

My final RMSE from the test set was 1.12. But for some reason, my random seed did not result in consistent sampling for the train test split, or consistent behavior from the models. However, after many iterations, the RMSE of this model consistently ranged from 1.09 to 1.14. So the number 1.12 may not be exactly correct, but should be close.