Movie Recommendation System

Lawrence Ejike

10/05/2021

1. Introduction

1.1. Background

Since the emergence of Netflix and in more recent years, there has been an increase in the number of online movie streaming platforms. Some examples include Amazon Prime, Disney Plus, Crave, and Apple TV, to name a few. A platform's ability to keep users engaged means more time spent by the users on the platform, which translates to longer periods of subscription by the user and more revenue for the platform providers. Thus, significant investments have been made into recommendation systems by companies that own and operate these platforms, as epitomized by the Netflix Prize competition of 2007-2009 that had a grand prize of US$1,000,000. Recommendation systems use ratings that users have given items to make a specific recommendation [1].

1.2. Executive summary

This report details the development of a movie recommendation system. The project utilized MovieLens 10M data set that was generated by GroupLens Research Lab. It contains over 10 million ratings for 10681 movies by 71567 users of the online movie recommender services, MovieLens.

The primary goal of this project was to create a movie recommendation system, one that uses ratings that users have given to movies to make specific predictions of the rating they would give other movies, thereby severing as a recommendation. The secondary goal of the project was to obtain root mean squared error, RMSE (a measure of the deviation of predicted values from observed values) that is less than or equal to 0.86490.

The development of the recommendation system was achieved using machine learning as well as other data science tools. The Movielens 10M data set was first split into two sets, validation and edx set, and the latter set was split into a train and test set. Predictive models using the train set were made that incorporate the effect of various features (movieId, userId, time/date, and genre) and these models were tested on the test set. The performance of each model was evaluated against each other and the target RMSE. Regularization was required in later models to offset the distortion arising from the difference in the number of ratings for each unique feature entry, thereby improving the performance of the models. After the target RMSE was obtained on the test set, the corresponding model was applied to the edx set and used to predict the ratings in the validation set, yielding a final RMSE value that met the target.

2. Method/Analysis

The following packages, tidyverse, caret, data.table, and lubridate were installed and used to download, clean, and create the data set.

i) Tidyverse was used for data wrangling, web-scrapping, joining, and reshaping data tables.

ii) Caret package was used for machine learning and building prediction algorithms.
iii) The data.table package was used as a fast file reader
iv) Lubridate package was used to simplify work with dates and time

2.1 Data Cleaning

2.1.1. Creating training (edx) set and hold out test (validation) set • The 10M MovieLens Data was downloaded as a ".DAT" file from the web and the rating data file was converted into a dataframe, named ratings. The four columns (variables) of the dataframe are userId for each user's unique identification, movieId representing each movie's unique ID number, rating for the score given to a movie by a user, and timestamp denoting the time of movie rating.

• 10M MovieLens data was also unpacked into a matrix called 'movies', and comprised of three columns named as 1)'movieId' representing each movie's unique ID number, 2)'title' representing each movie's title, and 3)'genres' representing each movie's genre type(s).

• The matrix, movies was converted to a dataframe (movies) and the components of the dataframe column movieId were converted to numeric, while those of columns, title, and genres were converted to character.

• A movielens dataframe was then created by joining the ratings and movies dataframes using the shared movieId column with a join function (left_join) that retained userId, movieId, rating, timestamp, title, and genre.

• The movielens data was then split using createDataPartition into edx sex (90% of the data) and validation set (remaining 10%). The validation set was updated using semi-join to ensure that userId and movieId it contained were also present in the edx set.

• The rows removed from the validation set were added back into the edx set, hence no data row in left out. The edx set was used as the training set while the validation set was held out for final predictions after the development of the final model.

2.1.2. Creating train set and test set from edx To be able to test the performance of the models being developed, the edx dataset was split into a train_set and a test_set. The train set was used for model building while the test set was used for prediction and performance checks. The method for creating the train and test set from the edx data set was similar to that for creating the edx and validation set from the original Movielens dataset.

The train and test sets were created using the createDataPartition function in the caret package. A temporary test set (test_set_pre) was created initially and then modified into a working test set (test_set). The latter was made by removing user and movie entries in the temporary test set that were not present in the train set.

The rows removed from the test set were added back into the train set, hence no data row in left out.

The metric used to assess the performance of each model to be built is residual mean squared error, RMSE. This represents the difference between values predicted by each model and the values observed. In related terms, it is the error made when predicting a movie rating. Hence, the lower the RMSE, the better the predictive power of a given model. The target RMSE for this project is equal to or less than 0.86490. The RMSE function used to assess each model is defined below. The number of digits to be printed was also set with the option function as shown below

2.2. Data Exploration, Data Visualization, and Modeling

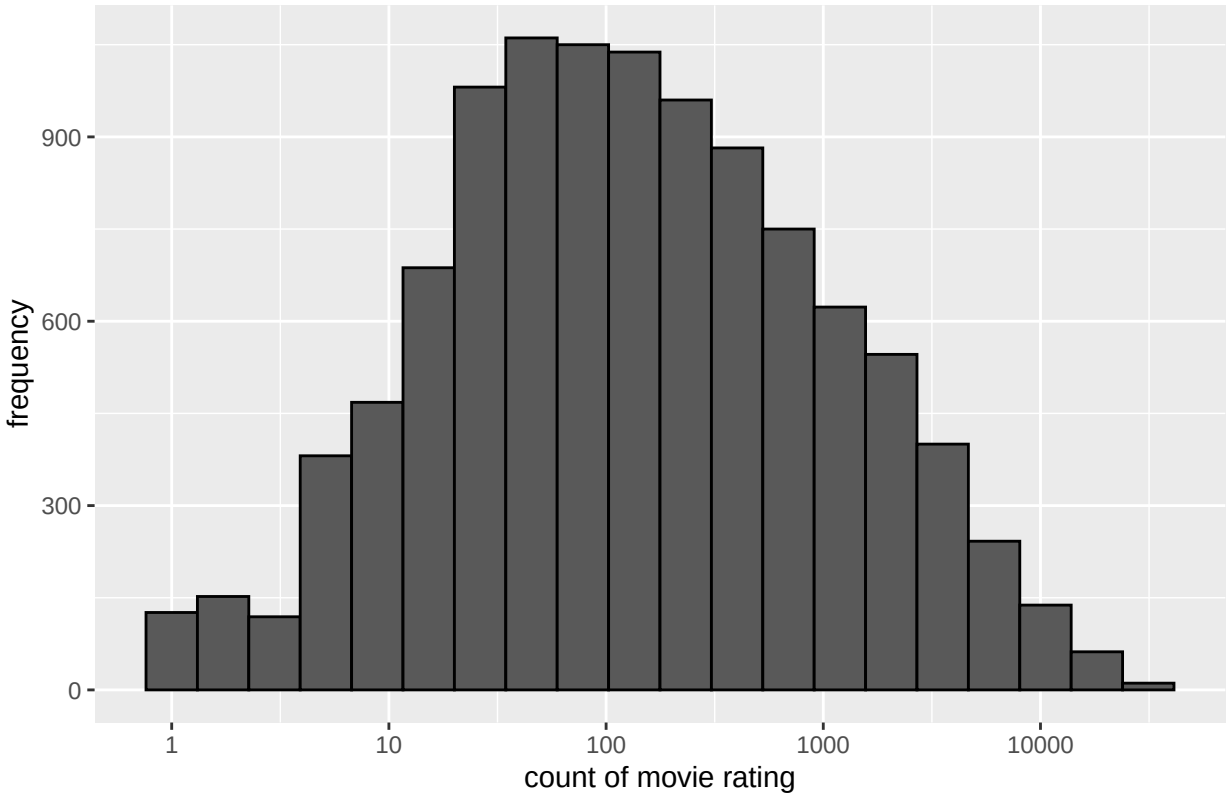
In this section, I explored the data in the data sets (edx and the train set) using tables and graphs where applicable and gained useful insights that formed the basis of subsequent modeling approaches. An examination of the edx data set shows it contains 9000055 rows and 6 columns or features (userId, movieId, rating,

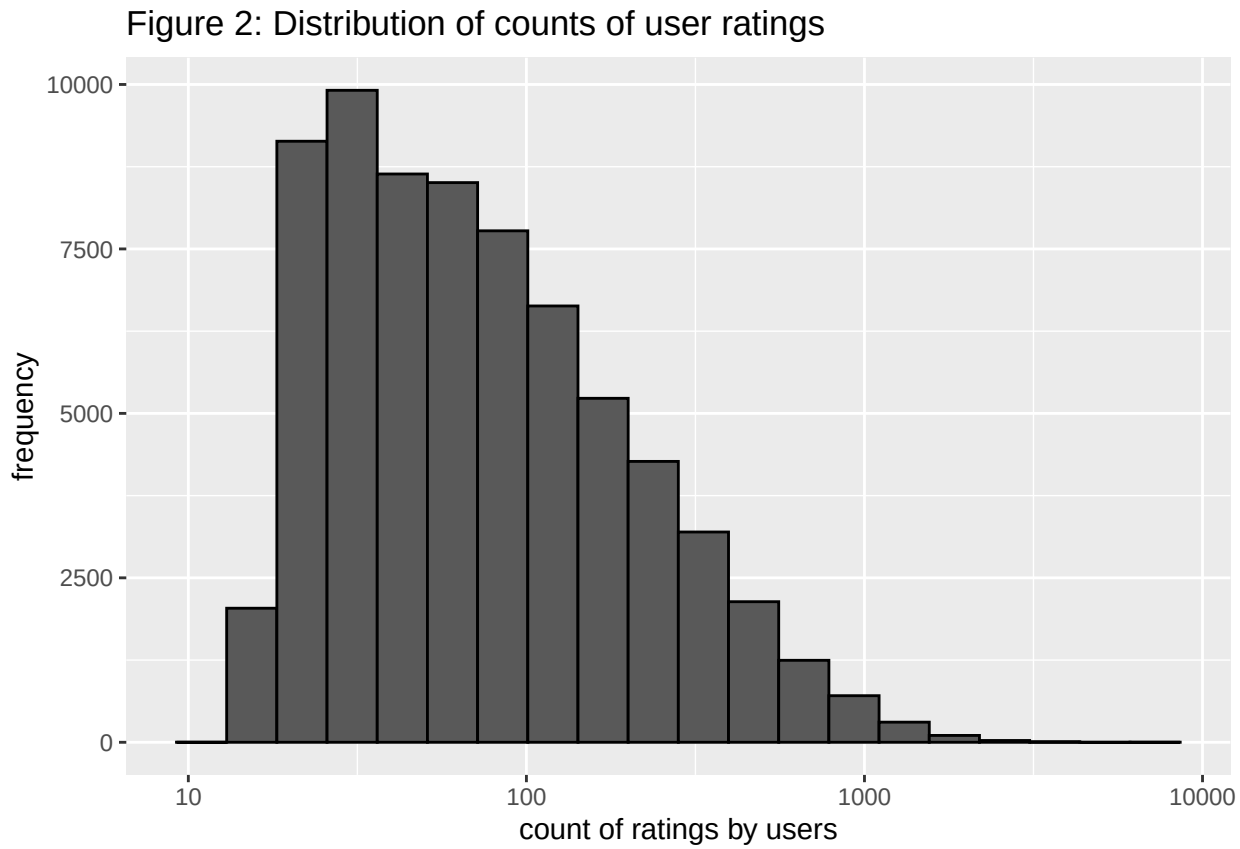
timestamp, title, and genres). As shown in Table 1 below, each row represented a unique combination of features. Since movieId and title represent the same information i.e. a given movie identity, I ignored the feature, 'title' so as not to duplicate information. The feature 'rating' represents the value for which predictions were to be made, and as such, were left out of consideration in the modeling approaches. Thus, the four features that were considered for this recommendation system were movieId, userId, timestamp, and genre. These features were explored sequentially, in the models presented below.

Table 1: Table 1: First 6 rows of the dataset, edx

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

Figure 1: Distribution of count of rated movies





Also, Figure 1 shows that some movies are rated more than others. Similarly, Figure 2 shows that some users rate more movies than other users. These differences in the number of ratings for unique features are accounted for in later models with regularization.

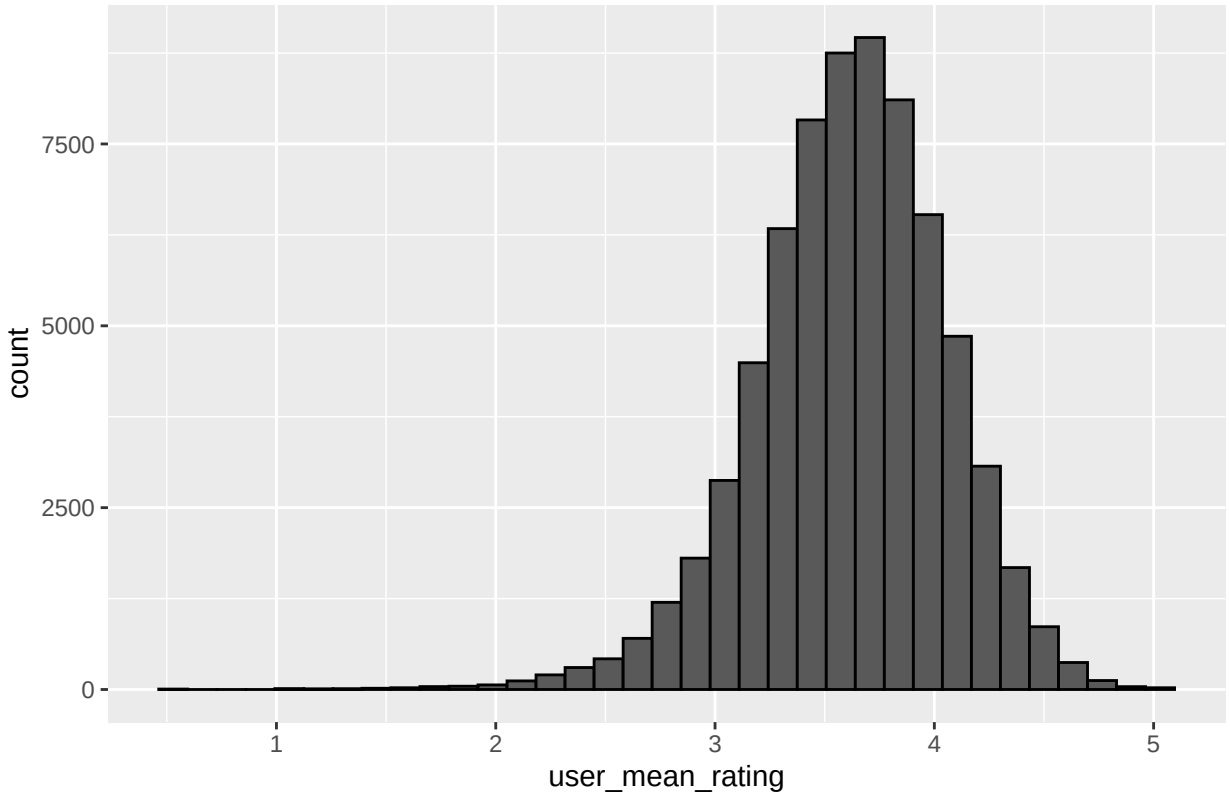
2.2.1. Model 0: Using the average Without deep introspection, the simplest model for predicting movie rating is one that gives the average of all movies as an estimate of a movie rating. This model assumes the same rating for all movies and users with the difference being explained by random error and is be expressed as rating = average rating + error or symbolically as Y_hat = mu + e. The estimated average rating, denoted here as mu was obtained as

The RMSE function was applied with mu as the predicted rating for all movie and user combinations. A table, rmse_results was also set up to which all models developed and their corresponding rmse values are stored for easy comparison.

2.2.2. Model 1 : Accounting for movie effects All movies are not rated the same. For example, some movies are blockbusters while others are not and consequently have lower ratings. The differences could stem from things like a particular movie cast, director, video quality, or particular fad or movie plot. This ultimately creates a bias with some movies being rated higher than others. Here, movie bias was introduced into the previous model and denoted as b_i, the average movie bias for movie i. The rating equation became rating = mu + b_i + e. Since there are millions of movies and obtaining the least square estimate of bi using lm() function is not practical, the b_i for each movie was computed as the mean of the differences between each movies rating and the average, mu (b_i = mean (rating- mu)) [1]. The b_i were computed using the train_set and predictions using the model, rating = mu + b_i were made on the test_set.

2.2.3. Model 2 : Accounting for user effects The average rating by each user was computed and the variability across users is shown by the histogram in figure 3.

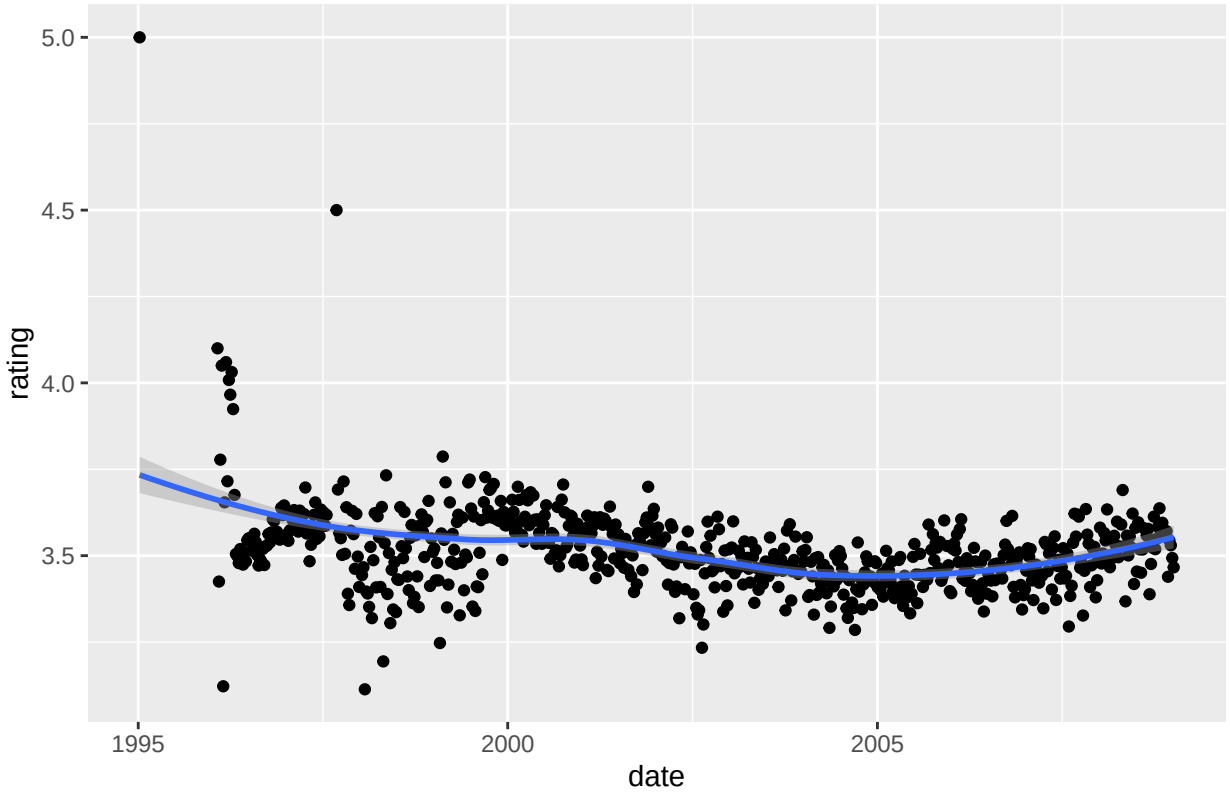
Figure 3: Distribution of average user ratings



The user-specific effect or user bias, b_u was then factored into the previous Model 1. As done earlier with movie bias, b_u was computed as follows: b_u = mean (rating- mu - b_i). The rating equation was updated to rating = mu + b_i + b_u + e, thereby incorporating both the movie and user bias.

2.2.4. Model_3: account for time effect(b_t) The timestamp column was first converted to a date column using lubridate package. The date could either be grouped by weeks, months, or years. As an example, a smooth plot of rating vs time(date) is shown in Figure 4 and this depicts the variability of rating, albeit marginally, across time. The time-specific effect or time bias, b_t was then factored into the previous model. The b_t was computed on the train_set as: b_t = mean (rating- mu - b_i - b_u), while predictions were made using rating = mu + b_i + b_u + b_t on the test_set. Three different models, 3a, 3b, and 3c were built to evaluate the effect of time bias in week, month, and year respectively and then, the best model was selected for further improvement.

Figure 4: Effect of time (in weeks) on rating



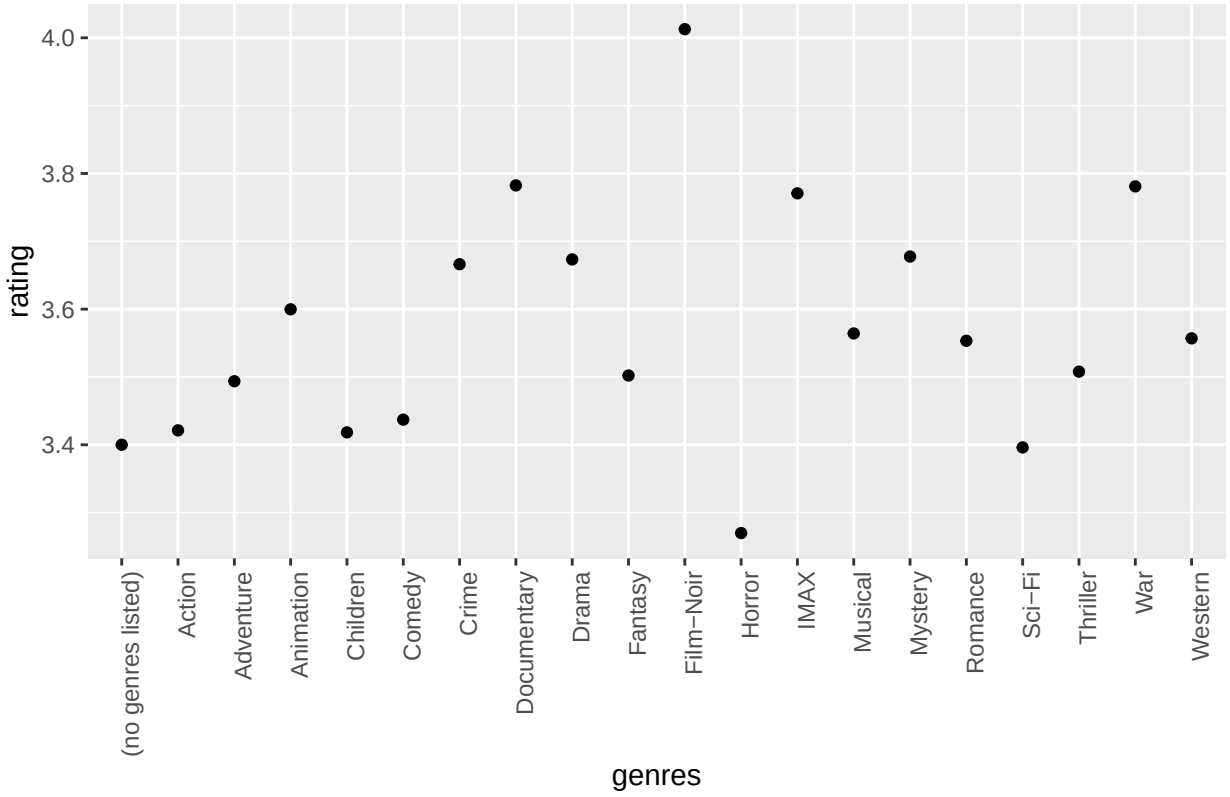
2.2.5. Model_4: account for multi-genre effect(b_mg) Movies belong to one or more genres. Table 2 below shows the number of unique genre present in the data, as well as the number of times it appears in the genre column and the average rating for the genre. Figure 5 is a scatterplot of rating vs genre, and it shows, as expected, that there are differences in genre rating. Some genres are just liked more than others and generally receive a comparatively higher rating. This bias due to genre contributes either positively or negatively to a movie's rating.

Table 2: Table 2: Average rating and count of unique genres in the train set

genres	count	rating
(no genres listed)	5	3.4000
Action	2048438	3.4212
Adventure	1527771	3.4936
Animation	373451	3.5997
Children	590356	3.4182
Comedy	2832980	3.4370
Crime	1061691	3.6662
Documentary	74348	3.7824
Drama	3128674	3.6732
Fantasy	740656	3.5021
Film-Noir	94736	4.0127
Horror	552622	3.2698
IMAX	6567	3.7707
Musical	346444	3.5641

genres	count	rating
Mystery	454655	3.6775
Romance	1369144	3.5533
Sci-Fi	1073697	3.3961
Thriller	1859514	3.5078
War	408949	3.7809
Western	151526	3.5569

Figure 5: Plot of genre vs average genre rating for train_set



For movies belonging to several genres, the total or multi-genre bias can be accounted for by summation of the individual bias of each constituent genre. Alternatively, the multi-genre bias can be obtained as the average rating of each unique genre combination as show in Table 3 below

Table 3: Table 3: Average rating and count of multi-genre in the train set

genres	count	rating
Drama	586799	3.7133
Comedy	561000	3.2382
Comedy|Romance	292546	3.4134
Comedy|Drama	259017	3.6007
Comedy|Drama|Romance	208892	3.6457
Drama|Romance	207106	3.6042
Action|Adventure|Sci-Fi	176134	3.5087
Action|Adventure|Thriller	119294	3.4360
Drama|Thriller	116103	3.4462

genres	count	rating
Crime|Drama	109916	3.9480
Drama|War	88996	3.9794
Crime|Drama|Thriller	84750	3.7800
Action|Adventure|Sci-Fi|Thriller	84206	3.5418
Action|Crime|Thriller	81695	3.4623
Action|Drama|War	79194	3.9170
Action|Thriller	77269	3.2730
Action|Sci-Fi|Thriller	76355	3.4192
Thriller	75719	3.5310
Horror|Thriller	59820	3.1153
Comedy|Crime	58952	3.5289

With the inclusion of multi-genre effect, the model rating equation became rating = mu + b_i + b_u + b_t + b_mg + e. The b_mg was computed on the train_set as: b_t = mean (rating- mu - b_i - b_u - b_t), while predictions were made using rating = mu + b_i + b_u + b_t + b_mg on the test_set.

2.2.6. Regularization

Table 4: Table 4: Movie bias and number of rating for the top 10 best movies

movieId	b_i	n
3226	1.4875	1
5194	1.4875	3
33264	1.4875	2
42783	1.4875	1
44653	1.4875	1
51209	1.4875	1
53355	1.4875	1
58898	1.4875	1
64275	1.4875	1
4454	1.4042	6

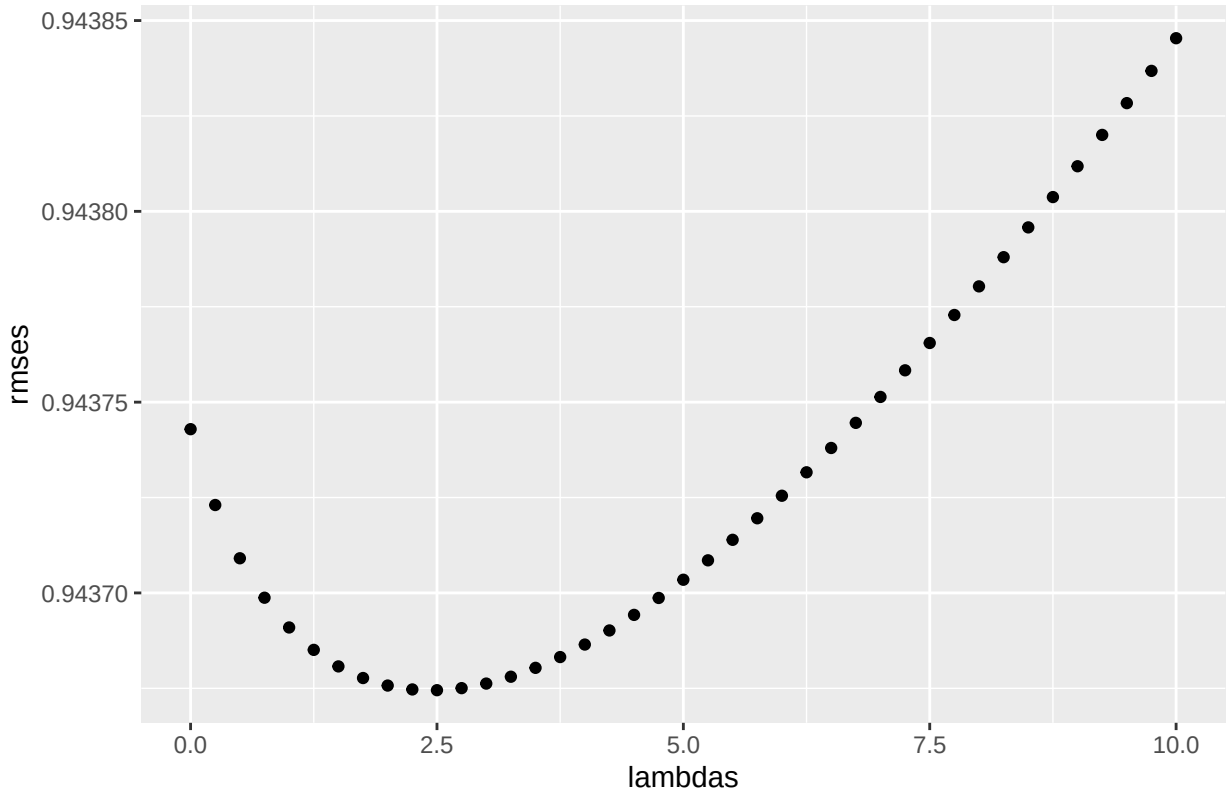
Table 5: Table 5: Movie bias and number of rating for the top 10 worst movies

movieId	b_i	n
5805	-3.0125	1
8394	-3.0125	1
61768	-3.0125	1
63828	-3.0125	1
64999	-3.0125	2
8859	-2.7500	40
6483	-2.6672	168
5837	-2.6375	4
61348	-2.6375	28
7282	-2.6034	11

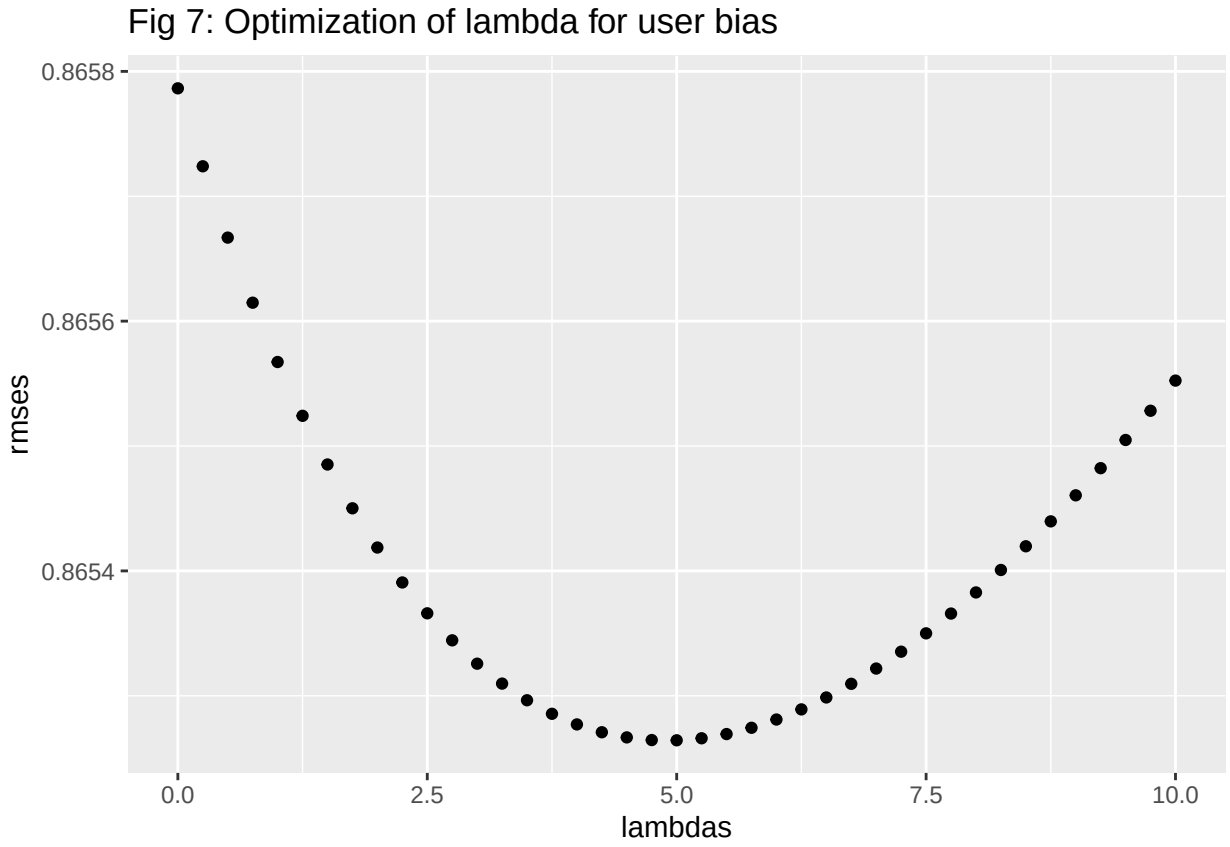
Some movies get rated more than others. As seen in tables 4 & 5 above, when we consider the best or worse rated movies, we see that these movies were rated by very few users. This small number of ratings results in greater uncertainty and larger estimates of the movie bias, whether positive or negative. We can minimize the distortion arising from such small sample sizes by applying regularization, which penalizes large estimates formed using small sample sizes [1]. In applying regularization, a regularized bias for a feature like movieId is obtained by dividing the calculated bias by a sum of the number of rating (n) for the movie i and a correction factor, lambda. This shrinks the bias of movies with a small number of ratings toward zero and leaves those with a large number of ratings unaffected. Lambda for each feature (movieId, userId, time, genre) bias was optimized over a range of values(0 to 10) to find which gives the minimum RMSE.

2.2.7. Model 5: Regularized movie effect model The optimized lambda value for movie bias was obtained by testing out different numbers in the range from 0 to 10 at 0.25 increments with the use of the sapply function. The lambda that gave the minimum value of RMSE in the optimization expression was chosen as the optimized lambda, lambda_i. This value, plugged back into the bias calculations, followed by subsequent prediction also gave the same RMSE value as the minimum RMSE as expected.

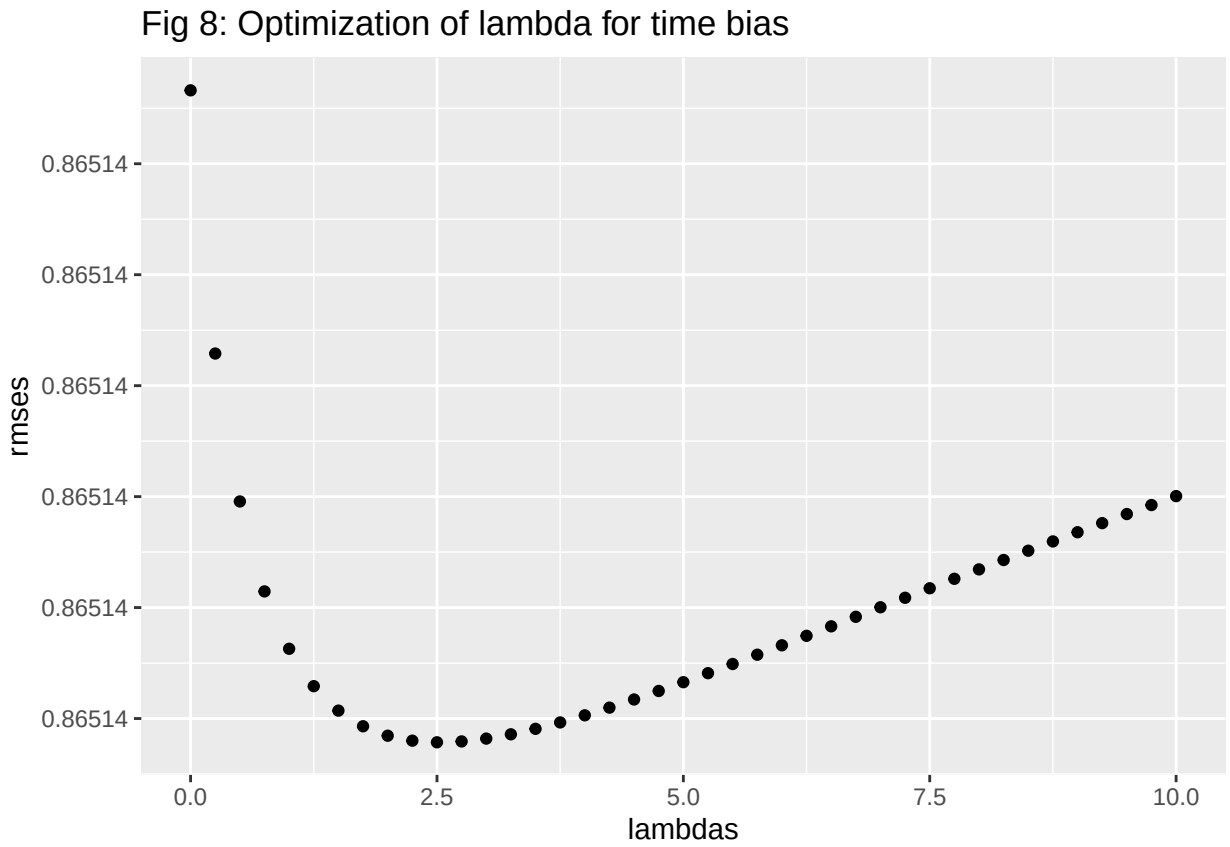
Fig 6: Optimization of lambda for movie bias



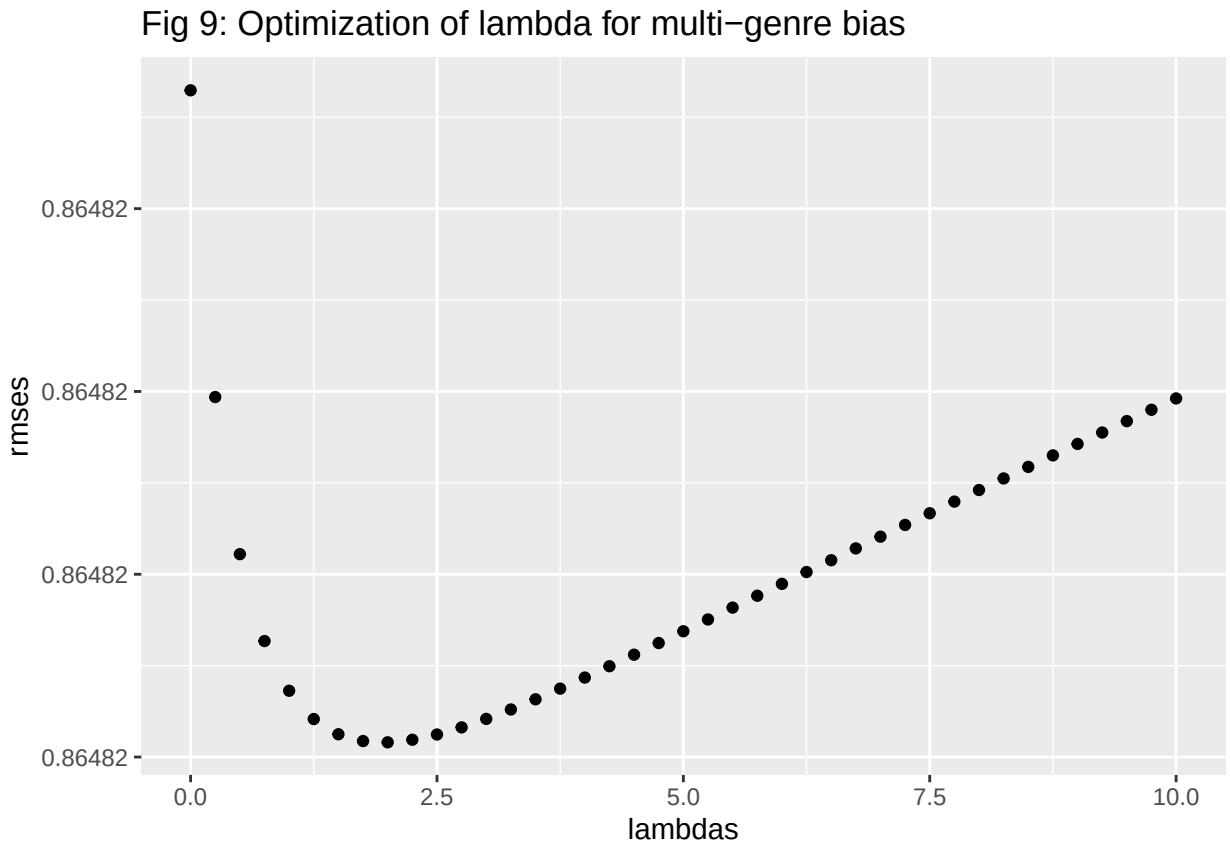
2.2.8. Model 6: Regularized movie + user effect model For Model 6, the user bias was regularized since some users rate more than others. With lambda_i fixed, the optimal lambda for user bias was obtained using the sapply function over the range 0 to 10 and saved as lambda_u. The minimum RMSE was assigned as the RMSE for Model 6.



2.2.9. Model 7: Regularized movie + user + time effect model For model 7, the time (in weeks) effect was incorporated into the previous model 6. Using the already obtained values of lambda_i and lambda_u, an optimized lambda for time effect (lambda_t) was obtained using sapply function over range 0 to 10 and The RMSE for this model corresponded to the minimum RMSE from the optimization



2.2.10. Model 8 : Regularized movie + user + time + multi-genre model Model 8 accounted for all features (movieId, user, time, and genre), much like model 4. For model 8, the multi-genre bias was factored into model 7. For optimization of lambda for multi-genre, all previously obtained lambdas for movies, users, and time were kept constant. Consequently, the RMSE for model 8 corresponded to the RMSE obtained with all optimized lambda values for the features.



2.2.11. Predicting ratings on the validation set Model 8 (Regularized movie + user + time + multi-genre model) was then formatted with the edx dataset as a train set and then used to make prediction on the validation set.

3. Result

Table 6: Table 6: RMSE results for different models

Approach	RMSE
Using the average	1.05990
Movie Effect Model	0.94374
Movie + User Effects Model	0.86593
Movie + User + time (week) Effects Model	0.86584
Movie + User + time (month) Effects Model	0.86591
Movie + User + time (year) Effects Model	0.86593
Movie + User + time + multi-genre Effects Model	0.86550
Regularized Movie Effects Model	0.94367
Regularized Movie + User Effects Model	0.86526
Regularized Movie + User + Time Effects Model	0.86514
Regularized Movie + User + Time + Multi-genre Effects Model	0.86482

Section 2.2 outlined the different modeling approaches for this movie recommendation system. The results for the models are displayed in the Table 6 above. The base model, Model 0 (Using the Average) predicted

the average rating of the train_set as the rating for entry in the test set. This gave an RMSE value of 1.05990, which was far greater than the target RMSE (<= 0.86490). Building on this model, Model 1 (Movie Effect Mode) incorporated the average movie bias and gave an RMSE of 0.94374. This was lower than the 'using the average' model but fell short of the target.

Model 2 (Movie + User Effects Model) improved on Model 1 by accounting for both movie & user bias, and this was reflected in the lower RMSE value of 0.86593. However, this value still did not meet the target RMSE. Model 3a, 3b, and 3c incorporated the effect of time in weeks, months, and year respectively into Model 2. In other words, in addition to accounting for movie and user bias as in Model 2, Model 3a accounted for the effect of the week of rating; Model 3b accounted for the month (but not week) of rating; while model 3c accounted for the year of the rating but not week or month. The RMSE for Model 3a, 3b and 3c were 0.86584, 0.86591 and 0.86593 respectively. As shown, Model 3a, which uses average time bias over weeks outperformed Model 3b (months) and 3c (years). It also performed better than the previous Model 2. Nevertheless, Model 3a still did not meet the target RMSE. In Model 4, the effect of genre (multi-genre) was accounted for as an addition to Model 3a. This lowered the RMSE value from 0.86584 to 0.86550, yet the latter was still higher than the target RMSE.

Having considered all features, regularization was then applied to offset the distortion present due to the discrepancy in the number of ratings between each unique member of features being considered. In Model 5 (Regularized Movie Effect Model), an optimized lambda of 2.5 was obtained for movie bias (as shown in Figure 6) and subsequently, an RMSE value of 0.94367. This RMSE was better than the non-regularized Model 1 but was far off the target still. For Model 6 (Regularized Movie + User Effects Model), with the lambda_i(movie) set as 2.5, an optimized lambda of 5 was obtained for lambda_u (user) as shown in figure 7. Model 6's prediction of rating on the test_set as indicated by its RMSE value of 0.86526 was much better than Model 5's predictions, as well as those of all preceding models. Notwithstanding, Model 6 did not meet the target RMSE.

Model 7 (Regularized Movie + User + Time Effects Model) introduced a regularization of the time bias (in weeks) as an addition to Model 6. Using previously obtained values of lambda_i and lambda_u, an optimized value of 2.5 (shown in figure 8) was obtained for lambda_t (time) as well as a corresponding RMSE of 0.86514. Although this was the lowest RMSE yet, it was still shy of the target. However, by introducing regularization of multi-genre bias in Model 8 (Regularized Movie + User + Time + Multi-genre Effects Model), I obtained an optimized value of 2 (shown in figure 9) for lambda_mg (multi_genre) and an RMSE of 0.86482 which met and exceeded the target RMSE(<= 0.86490).

Using model 8 updated with the edx set which contained all the movieId and UserId present in the validation set,ratings were predicted on the validation set and an RMSE value of 0.86434 was obtained when compared to the observed ratings. This RMSE value met and exceeded the target RMSE(<= 0.86490) for the project.

4. Conclusion

This report outlines the development of a recommendation system that predicts the rating that unique users give unique movies. The project was built on the Movielens 10M data set which contained over 10 million ratings as well as 10681 movies and 71567 users. A secondary goal for this project was to obtain a target RMSE equal to or less than 0.86490 based on prediction. Initially, several prediction models (Model 1 to 4) were developed that started with the average rating as a base before factoring the effect of movie, user, time, and multi-genre biases sequentially. With these models failing to meet the target RMSE, regularization was applied to the models to give new models (Model 5 to 8) with improved RMSE compared to the corresponding non-regularized models (Model 1 to 4). The best model with the lowest RMSE was a regularized model that accounted for the movie, user, time, and multi-genre biases (Model 8). This model was applied to the edx data set and used to make predictions on the validation set, yielding an RMSE of 0.86434 that met the target RMSE(<= 0.86490) for the project, making the project a success.

One major limitation of the current regularization method employed in this project is that it does not account for variations arising from the fact that (i) some groups of movies have similar rating patterns and (ii) groups of users have similar rating patterns as well[1]. For the future, improvements on this model should

also apply matrix factorization which will account for the aforementioned variations. Matrix factorization would involve converting the data set into a matrix in which each movie gets a column and each user gets a row [1]. Other improvements could also include breaking down multi-genre into unique genres prior to accounting for the genre effect. This will also allow effective factoring of genre for a new movies with new combination of genres than already exist in our data set.

Reference

[1] Irizarry R. I., Introduction to Data Science. 2019