Movie Recommendation

Contents

1 Introducton **2**

2 Data preparation **2**
2.1 Data Description 2

3 Data Exploration and Visualization **3**
3.1 Movie Rating 3
3.2 User Rating 3
3.3 Release Year Rating 4

4 Modelling **4**
4.1 Model 1: Just the Average 4
4.2 Model 2: Average + Movie Effect 5
4.3 Model 3: Average + Movie + User Effect 6
4.4 Model 4: Average + movie + user + year effect 6
4.5 Final Test 7

5 Conclusion **7**

1 Introducton

All popular movie websites or streaming services like Netflix apply recommendation systems to suggest movies to viewers. Netflix in particular uses a recommendation system that predicts how many stars a user will give a specific movie. One star is the lowest, and five the highest. In 2006 Netflix offered 1 million dollars to the team that could improve the current recommendation system by 10%. The challenge decided on a winner based on the residual mean squared error (RMSE). The goal of this project is to create a movie recommendation system using the Movie Lens data set that results in a RMSE of less than 0.86490.

2 Data preparation

2.1 Data Description

MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota over various periods of time, depending on the size of the set. The 10M version of the MovieLens data set was used for this project which includes 7 variables: userId, movieId, rating, timestamp, title, genres and year ("year" was separated from the "title" variable to represent release year for a specific movie). The data set was also split into a train set (edx, 90%) and test set (final_holdout_test, 10%). edx will be used to train the final model, and final_holdout_test set will be used to test it. We start by exploring the overall nature of the edx set by examining the structure. The summary of this data set is presented in Table 1 and Table 2.

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  7 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang " "Net, The " "Outbreak " "Stargate " ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  $ year     : num  1992 1995 1995 1994 1994 ...
```

Table 1: Summary of Character Variables

Character_variables	Number_of_categories
title	10407
genres	797

Table 2: Summary of Numeric Variables

Numeric_variables	Min	Median	Mean	Max	NA_number
userId	1.0	35738	35869.8	71567	0
movieId	1.0	1834	4121.7	65133	0
rating	0.5	4	3.5	5	0
timestamp	789652009.0	1035493918	1032615907.4	1231131736	0
year	1915.0	1994	1990.2	2008	0

From this summary, we see that there are 7 variables, including 2 character and 5 numeric. Since the objective is to build a model that predicts rating, rating will be our response variable, and the remaining variables will be the predictors. It is important to note that the rating variable has a range of 0.5 to 5. This makes sense because it is based on a 5 star scale and we do not want to see values outside this range. Also note that the mean value for all ratings is 3.5. Furthermore, we can see that this data set is already in tidy format with no NA's, thus, no further cleaning is required.

3 Data Exploration and Visualization

3.1 Movie Rating

This section explores the ratings given to specific movies. Specifically, the average rating per movie, and the number of ratings per movie. To get an idea of the distribution, we take a look at the histogram of each in figure 1.

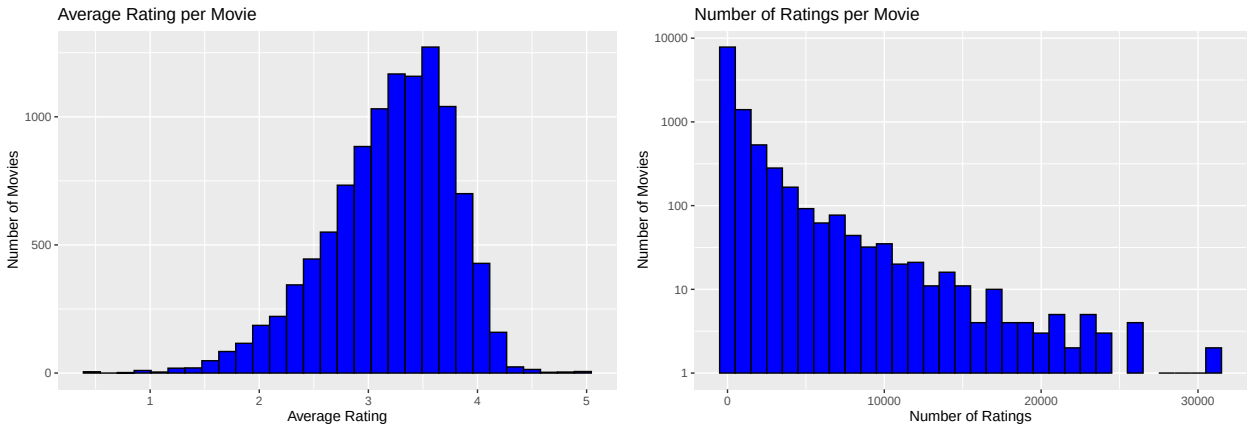

Figure 1: Histograms of Movie Ratings

There are 10,677 different movies in this data set, with rating counts ranging from 1 to 31,362. This means that some movies have been rated only once, while others have been rated more than 31,000 times! Additionally, since some movies are generally more liked than others, we see a large variation in average rating as well. This implies that our movie recommendation model needs to account for this bias, which we will denote as the "Movie Effect". The Movie Effect should also contain some sort of penalization to account for the wide range of ratings per movie.

3.2 User Rating

Next, we will explore the ratings given by specific users. Specifically, the average rating given by each user, and the number of ratings given per user. Again, we take a look at the histograms in figure 2 to get a sense of the distribution.

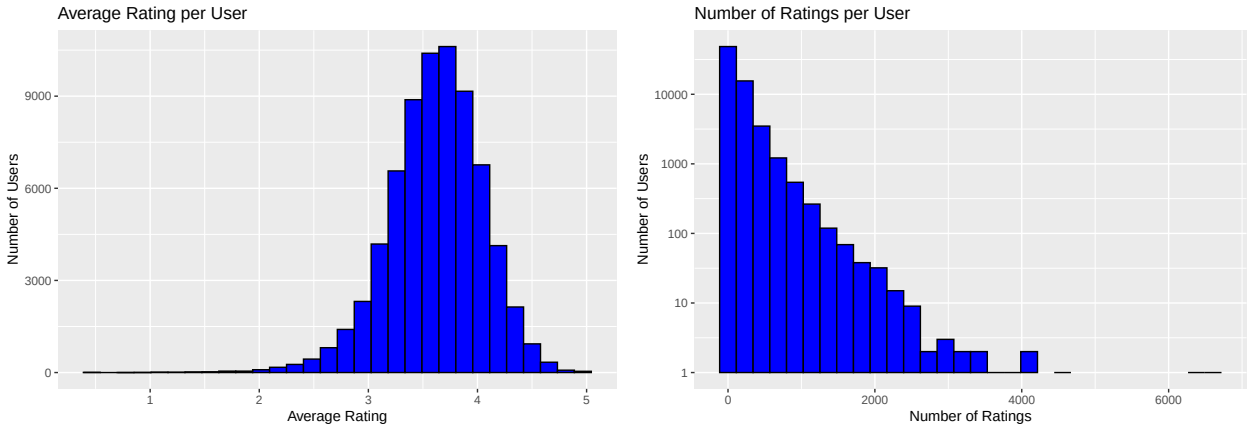

Figure 2: Histograms of User Ratings

There are 69,878 different users in this data set, with rating counts ranging from 10 to 6,616. This means

that some users have only rated 10 movies, while others have rated almost every movie in the data set! We can also see from the average rating distribution that some users generally enjoy movies and give a higher rating on average, while others are far more critical and give lower ratings on average. This implies that our model should also include a penalized "User Effect" to account for this variability.

3.3 Release Year Rating

As mentioned earlier, the "year" variable was extracted from the "title" variable to represent release year for each movie in the data set. Following the same process as with the movie and user variables, we look at the average rating per year and the number of ratings given each year. This time however, line plots are used to show the change over time.

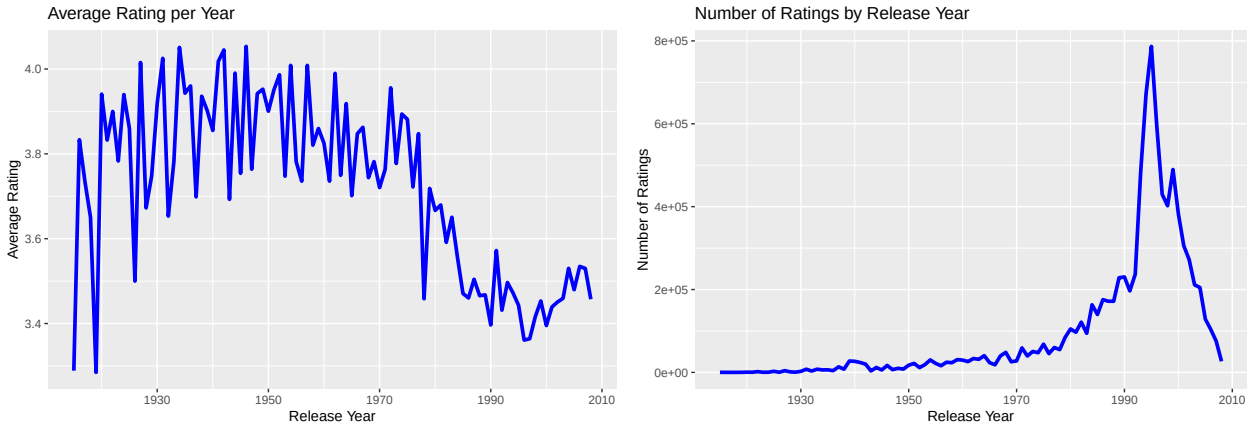

Figure 3: Line Plots of Realease Year Ratings

The range of number of ratings for release year is 32 to 786,762. It can be seen from Figure 3 that the number of ratings increases gradually from 1915 to 1990, but soars significantly from 1990 to 1995 and peaks at 1995. After that, the number drops from 1995 to 2008. One should also note that average rating was quite volatile over the years. Again, this implies that the recommendation model should account for these biases with a penalized "Year Effect".

4 Modelling

We are now ready to start building a regression model for the Movie Recommendation System. As stated earlier, the efficacy of the model will be evaluated using the residual mean squared error (RMSE). It is the typical error we make when predicting a movie rating. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{Y}_{u,i} - Y_{u,i})^2} \tag{1}$$

where $Y_{u,i}$ is the rating for movie i by user u, $\hat{Y}_{u,i}$ is the predicted value of $Y_{u,i}$ and N is the number of user/movie combinations. As a reminder, the goal of this project is to attain a RMSE less than 0.86490. Also note that the models that follow use the edx data set that was partitioned further into two sets: 90% of the data was used to train the model, and the remaining 10% was used as a test set to calculate the RMSE. Once we find a model that works, we will retrain using the edx set and test on the final_holdout_test set.

4.1 Model 1: Just the Average

As a starting point, we create a simple model that uses only the overall average rating. In other words, for all user/movie combinations, the predicted rating is just the average rating of the whole data set.

Model 1

$$\hat{Y}_{u,i} = \mu + \epsilon_{u,i} \tag{2}$$

where $\epsilon_{u,i}$ is an independent and identically distributed error centered at 0 and μ is the mean rating of all movies. The result of this model can be seen in Table 3

Table 3: Result of Model 1

Model	RMSE
Average Rating	1.060054

This model gives a RMSE of 1.06, which is larger than the desired result. Therefore, the average alone is not a good enough predictor, and we must add more variables to the model.

4.2 Model 2: Average + Movie Effect

During data exploration, it was discovered that the average rating per movie varied. This implies that our model should incorporate a "Movie Effect".

Model 2

$$\hat{Y}_{u,i} = \mu + b_i + \epsilon_{u,i} \tag{3}$$

where the term b_i was added to represent the movie effect. We can estimate b_i by taking the average of $\hat{Y}_{u,i}$ - μ.

$$\hat{b}_i = \frac{1}{N_i} \sum_i (\hat{Y}_{u,i} - \mu) \tag{4}$$

It was also discovered that some movies were rated very few times while others were rated many times. Since we have to divide by N_i in order to get $\hat{b}_i$, the estimates with many ratings will be more precise than those with fewer ratings. Therefore, we can use regularization, or a penalized regression model. We can control the total variability of the movie effect by minimizing an equation that adds a penalty:

$$\frac{1}{N} \sum_{u,i} (Y_{u,i} - \mu - b_i)^2 + \lambda \sum_i b_i^2 \tag{5}$$

The first term is just the sum of squares and the second is a penalty that gets larger when many b_i are large. The values of b_i that minimize this equation are:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{i=1}^{n_i} (Y_{u,i} - \hat{\mu}) \tag{6}$$

Now, when n_i is small, the estimate $\hat{b}_i$ reduces toward 0. The larger λ, the smaller it gets. And so the new model is:

$$\hat{Y}_{u,i} = \mu + \hat{b}_i(\lambda) + \epsilon_{u,i} \tag{7}$$

To select the optimal λ, we can use cross validation, and use the value that minimizes RMSE. Table 4 shows the results.

Table 4: Result of Model 2

Model	Optimal_lambda	RMSE
Movie Effect	1.6	0.9429369

This model gives a RMSE of 0.943, which is still larger than the desired result. Therefore more variables are required.

4.3 Model 3: Average + Movie + User Effect

Next we will try adding a "User Effect" to the model to account for the fact that some users tend to rate movies low, while others rate movies high.

Model 3

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i} \tag{8}$$

where the term b_u was added to represent the user effect. We can estimate the penalized b_u just like before:

$$\hat{b}_u(\lambda) = \frac{1}{(\lambda + n_u)} \sum_{i=1}^{n_u} (Y_{u,i} - \hat{\mu} - \hat{b}_i(\lambda)) \tag{9}$$

where n_u is the number ratings given by user u. Again, we use cross validation to select the optimal lambda and use the value that minimizes RMSE. Table 5 summarizes the results.

Table 5: Result of Model 3

Model	Optimal_lambda	RMSE
Movie + User Effect	0.2	0.9415197

This model provided a RMSE of 0.942, which is lower, but not low enough. Let's add another variable.

4.4 Model 4: Average + movie + user + year effect

It could be seen that due to variability in both average rating and number of ratings for release year, our model should contain a "Year Effect" as well. Thus,

Model 4

$$Y_{u,i} = \mu + b_i + b_u + b_y + \epsilon_{u,i} \tag{10}$$

where the term b_y was added to represent the year effect. Since the method to obtain b_y is the same as above, let's skip right to the results. Table 6 shows the results for all methods for comparison.

Table 6: Combined result of all models

Model	RMSE	Optimal_lambda
Average Rating	1.0600537	NA
Movie Effect	0.9429369	1.6
Movie + User Effect	0.9415197	0.2
Movie + User + Year Effect	0.8638376	4.6

We can see that with a RMSE of 0.86384, model 4 provides the desired result.

4.5 Final Test

Now that we found a model that works, we will retrain the edx set using the same methodology: a penalized least squares regression model that incorporates movie, user, and year effects. Then use it on the final_holdout_test set to get the final RMSE. See Table 7.

Table 7: Final RMSE Result

Model	Optimal_lambda	RMSE
Movie + User + Year	4.9	0.8645218

The final RMSE on the final_holdout_test set is lower than the desired result. Hence, we can conclude that the movie recommendation system works effectively, and provides a decent prediction for the rating of a specific movie, given by a specific user.

5 Conclusion

Every movie website and streaming service uses a movie recommendation system in order to enhance user enjoyment. The more accurate the prediction, the more enjoyment for the user. This project used the MovieLens data set to build a model using a penalized least squares approach and attained a RMSE of 0.86452, which is lower than the goal of 0.86490. But not by much. The RMSE could be reduced even further by implementing more effects such as genre, timestamp, user's age, user's sex, etc. Additionally, other machine learning models could also be used to achieve better results when there are no limitations in processing power and RAM. But for the purposes of this project, our model will suffice.