MovieLens Recommendation System - Capstone

Martinez Maldonado Matías

Last compiled on marzo 05, 2023

Contents

1 Introduction

Recommendation engines are a subclass of machine learning which generally deal with ranking or rating products / users. Loosely defined, a recommender system is a system which predicts ratings a user might give to a specific item (a movie). These predictions will then be ranked and returned back to the user.

They're used by various large name companies like Google, Instagram, Spotify, Amazon, Reddit, Netflix etc. often to increase engagement with users and the platform.

The version of movielens that we will use contains around 9 Millions movie ratings for train my model, which implies that we will divide this 9 Millions movie ratings into train and test set. And to validate the model we will use *final_holdout_test* with around 6.7 Millions movie ratings.

The purpose of this project is creating a recommendation system using the **MovieLens dataset** that beats a RMSE less than 0.86490.

This report sets out the exploratory analysis of the data using common visualization techniques followed by the methods used to develop, train and test the algorithm before providing and discussing the results from each iteration of the algorithm and concluding on the outcome of the final model and its limitations.

2 Pre-Processing Data

First of all, we pre-processing the data to manipulate it easier. Also, we pre-process data to handle missing values and extract some important variables which are inside features.

2.1 Searching for missing values.

As we see, there is no missing values. This is really important because now we skip steps due to we don't have to fill this missing values.

Table 1: Missing Values

	x
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

2.2 Creating a long version of both validation and train data set separating *genres* by row

```
edx_temp <- edx %>%
  separate_rows(genres, sep = "\\|", convert = T)


final_holdout_test <- final_holdout_test %>%
  separate_rows(genres, sep = "\\|", convert = T)
```

2.3 Converting *timestamp* predictor into a human most readable format.

```
edx_temp$year <- edx_temp$timestamp %>% as_datetime() %>% year()
edx_temp$month <- edx_temp$timestamp %>% as_datetime() %>% month()
```

2.4 Extract the *release date* from *title.*

```
# Extract the release date from title to a new predictor
edx_temp <- edx_temp %>% mutate(release_date = title %>% str_extract_all("\\([0-9]{4}\\)") %>%
                 str_extract("[0-9]{4}") %>% as.numeric(),
               title = title %>% str_remove("\\([0-9]{4}\\)")%>% str_trim("right"))

# Doing the same with the validation dataset in one step
final_holdout_test <- final_holdout_test %>% mutate(release_date = title %>% str_extract_all("\\([0-9]{
                               str_extract("[0-9]{4}") %>% as.numeric(),
                             title = title %>% str_remove("\\([0-9]{4}\\)")%>% str_trim("right"),
                             year = timestamp %>% as_datetime() %>% year(),
                             month = timestamp %>% as_datetime() %>% month())
```

Also we select the most important features in both datasets.

3 Data Exploration

After pre-processing the data, we start the exploratory data analysis (EDA) to complain how best to manipulate data sources to get the answers we need, making it easier for us to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

Then we ask ourselves a series of questions. This questions are related with How many users are? How many movies are rated? Which are the top 20 most rated movies? What is the frequency of ratings? etc.

First of all, we take a look at the dataset structure and make a statistical summary of the features that are into it.

3.1 Structure and Summary

```
## tibble [23,371,423 x 8] (S3: tbl_df/tbl/data.frame)
##  $ userId      : int [1:23371423] 1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId     : int [1:23371423] 122 122 185 185 185 292 292 292 292 316 ...
##  $ rating      : num [1:23371423] 5 5 5 5 5 5 5 5 5 5 ...
##  $ title       : chr [1:23371423] "Boomerang" "Boomerang" "Net, The" "Net, The" ...
##  $ genres      : chr [1:23371423] "Comedy" "Romance" "Action" "Crime" ...
##  $ year        : num [1:23371423] 1996 1996 1996 1996 1996 ...
##  $ month       : num [1:23371423] 8 8 8 8 8 8 8 8 8 8 ...
##  $ release_date: num [1:23371423] 1992 1992 1995 1995 1995 ...
```

```
##      userId         movieId          rating         title          
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:23371423   
##  1st Qu.:18140   1st Qu.:  616   1st Qu.:3.000   Class :character  
##  Median :35784   Median : 1748   Median :4.000   Mode  :character  
##  Mean   :35886   Mean   : 4277   Mean   :3.527                     
##  3rd Qu.:53638   3rd Qu.: 3635   3rd Qu.:4.000                     
##  Max.   :71567   Max.   :65133   Max.   :5.000                     
##     genres               year          month         release_date 
##  Length:23371423    Min.   :1995   Min.   : 1.000   Min.   :1915  
##  Class :character   1st Qu.:2000   1st Qu.: 4.000   1st Qu.:1987  
##  Mode  :character   Median :2003   Median : 7.000   Median :1995  
##                     Mean   :2002   Mean   : 6.789   Mean   :1990  
##                     3rd Qu.:2005   3rd Qu.:10.000   3rd Qu.:1998  
##                     Max.   :2009   Max.   :12.000   Max.   :2008  
```

The features in both datasets are:

- **userId** `<integer>` that contains the unique identification number for each user.
- **movieId** `<integer>` that contains the unique identification number for each movie.
- **rating** `<numeric>` that contains the rating of one movie by one user. Ratings are made on a 5-Star scale with half-star increments.
- **title** `<character>` that contains the title of each movie including the year of the release.
- **genres** `<character>` that contains a genre of each movie.
- **year** `<numeric>` that contains the year of each rating.
- **month** `<numeric>` that contains the month of each rating.
- **release_date** `<numeric>` that contains the year of each movie release date.

3.2 Movies Vs Users

Table 2: Number of Movies vs Number of Users

n_users	n_movies
69878	10677

If we see this table and multiply this two values, we could realize that not every movie is rated and not every user rate every movie.

3.3 Frequency Rating

Table 3: Frequency Rating

rating	count
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

In this table, we see the frequency distribution of ratings. As we see, the most frequent rating is four stars and the less one is a half star. To compare this values in a easier way, let's make a plot.

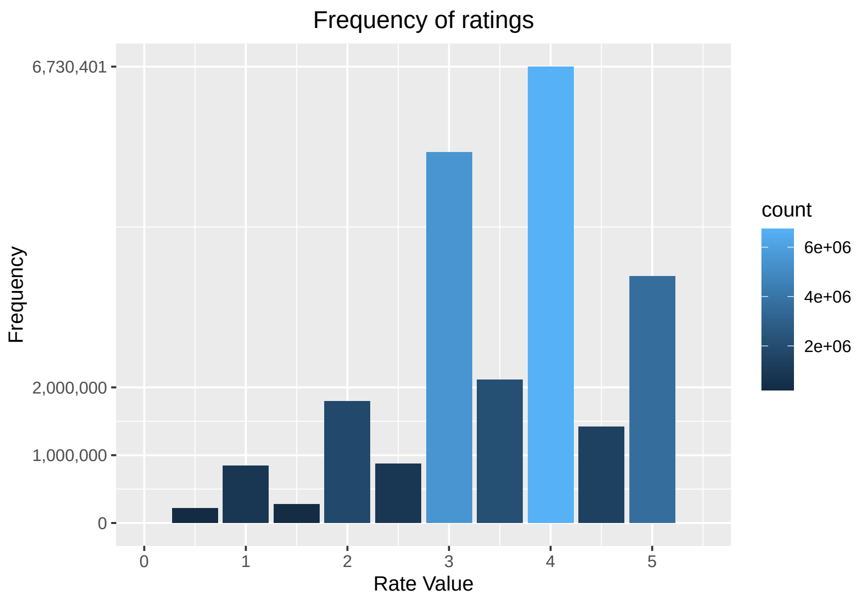

Figure 1: Frequency of Ratings

3.4 Release Date Plots

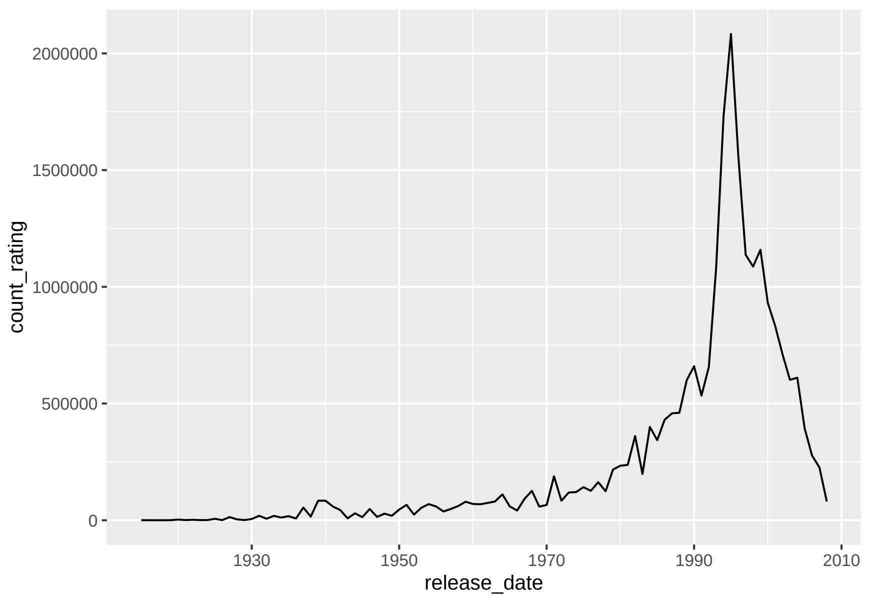

Figure 2: Release Date Distribution

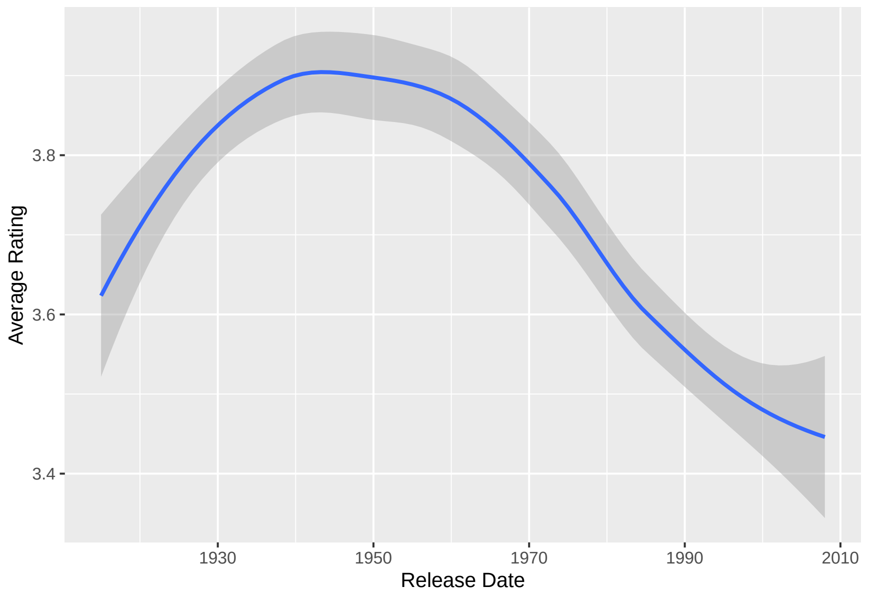

Figure 3: Average Rating by Release Date

In Fig. 2 see the release date distribution of every movie in the dataset. We clearly notice that the most movies are released after 1990 and the releases peak was on 1995 approximately.

In order to explore the effect of the year of release on average rating, we make a average rating by release date plot (Fig. 3). As we can see effectively average rating varied by year of release. Interestingly, the average rating peaked for movies released between 1940 and 1950 and declined for movies released since that period.

3.5 Top 20 Most/Less Rated Movies

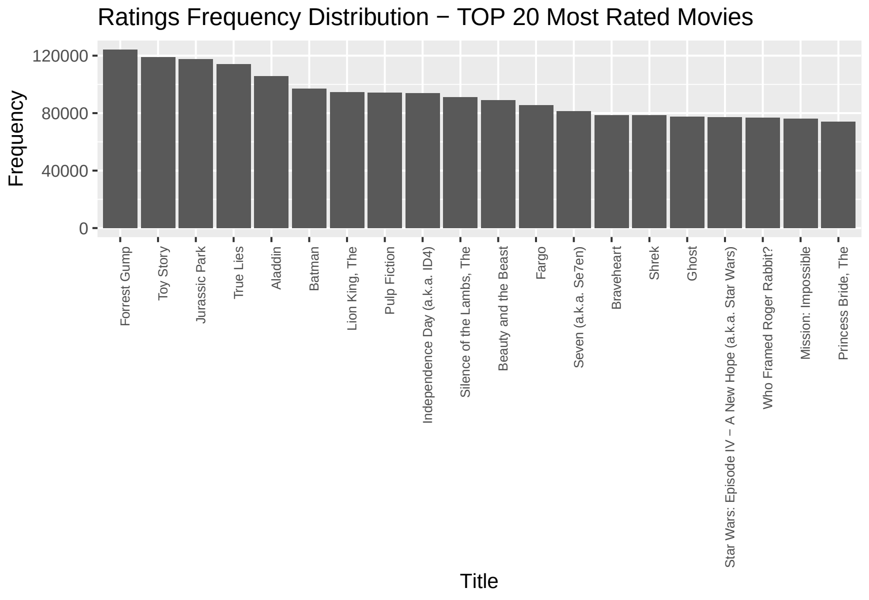

Figure 4: Top 20 Most Rated Movies

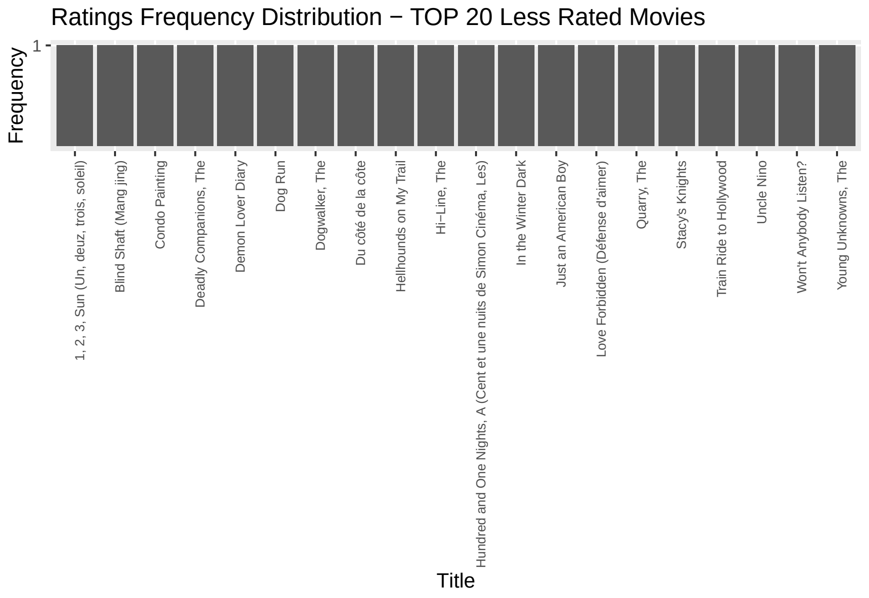

Figure 5: Top 20 Less Rated Movies

The last two plot show us the both top 20 most and less rated movies in this dataset. The most rated movie is *Pulp Fiction* followed by *Forrest Gump* and *The Silence of the Lambs*.

In the other hand, the top 20 less rated movies has the same rated frequency with just one rating. Despite this we could name three movies in this top 20: *The Young Unknowns*, *Won't Anybody Listen?* and *When Time Run Out... (a.k.a. The Day the World Ended)*.

3.6 Popularity Effect

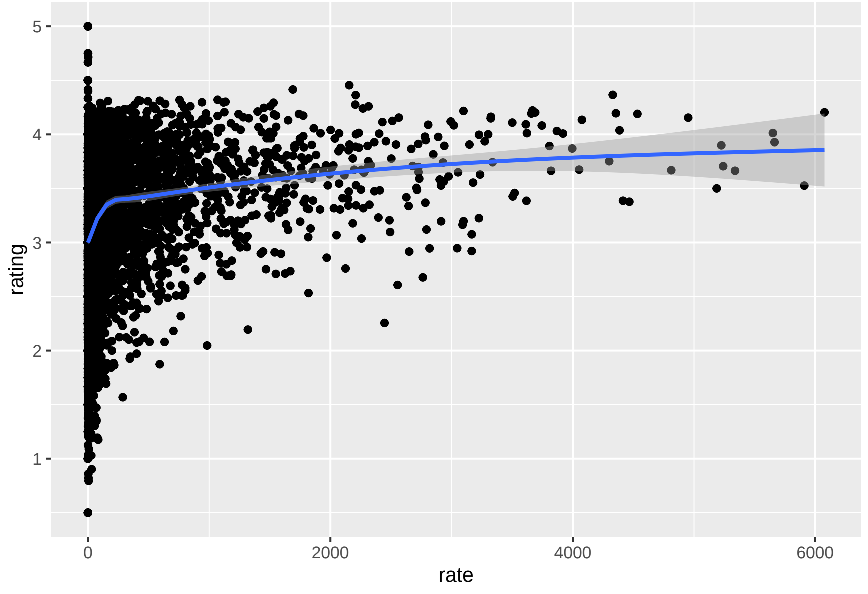

Figure 6: Trend Movie Ratings

We made a plot (Fig. 6) about the number of times a movie is ranked per year and the average number of stars that movie has. We find out a popularity effect: the more often a movie is rated, the higher its average rating.

3.7 Number of Ratings for Each Genre

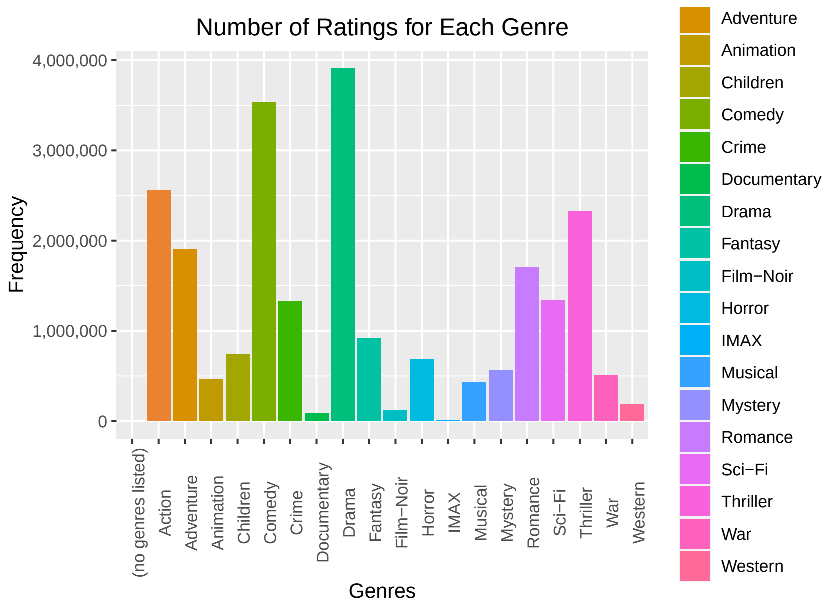

Figure 7: Number of Ratings for Each Genre

Table 4: Number of Ratings for Each Genre

genres	count
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332
War	511147
Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7

Here, we see the frequency distribution of ratings by genre. As we see, the most rated genre is *Drama* followed by *Comedy* and *Action*.

3.8 Median and Mean Ratings by Genre

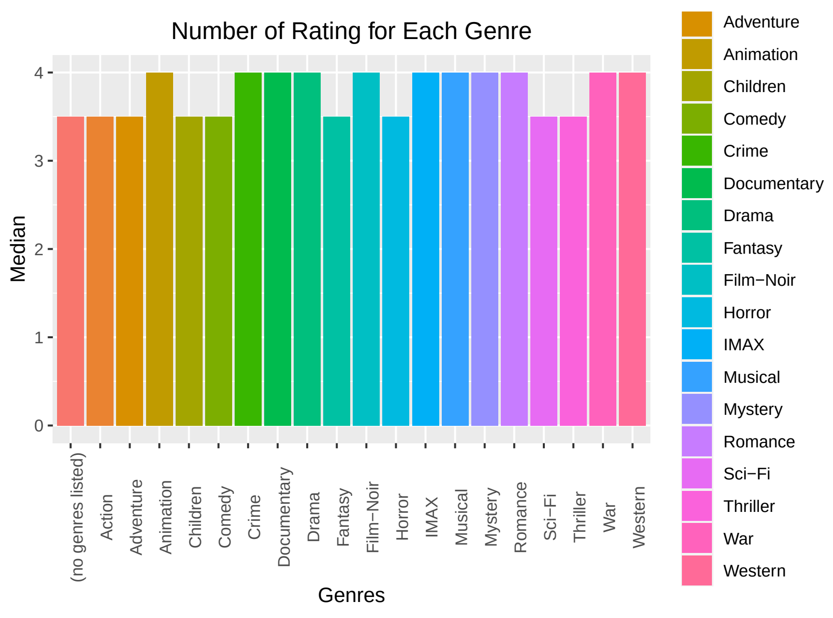

Figure 8: Median Ratings by Genre

Here we plot the median instead of mean because we consider more robust. And as we see, all genres ratings range from 3.5 and 4.

And then we plot (Fig. 9) the average rating by genres and his error only if the genre has more than 1000 ratings.

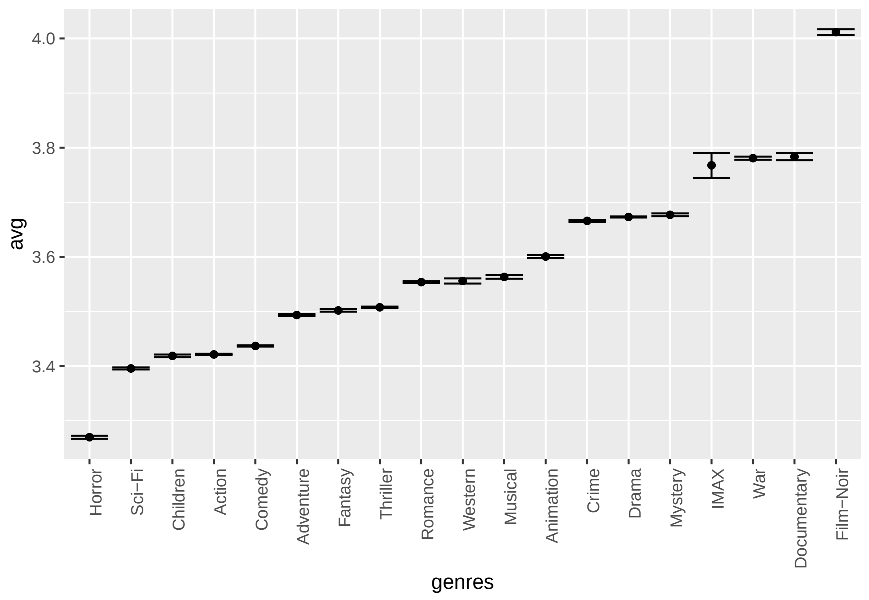

Figure 9: Mean Rating by Genre (genres with more than 1000 ratings)

4 Creating the Model

4.1 Creating data partition

As the final_hold_out set was reserved for final validation, the edx dataset needed to be used both to train and test the algorithm in development. Hence, we need to choose use cross-validation or split the edx dataset into train_set and test_set to train our algorithm. We choose the second one.

We use the *edx_temp* dataset created in the pre-processing section. The *test_set* contains 25% of the edx_temp data.

4.2 Loss function

The Residual mean square error (RMSE) is defined as the standard deviation of the residuals where residuals are a measure of spread of data points from regression line. In other words, it tells you how concentrated the data is around the line of best fit. The RMSE was calculated to represent the error loss between the predicted ratings derived from applying the algorithm and actual ratings in the test set. The RMSE is then defined as:

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

We define $Y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{Y}_{u,i}$.

4.3 Let's start with a naive approach (Just the average)

We know that the estimate that minimizes the RMSE is the least squares estimate of $\hat{\mu}$ and, in this case, is the average of all ratings. Let's start by building the simplest possible recommendation system. In this, we predict the same rating for all movies.

The formula is this:

$$Y_{u,i} = \hat{\mu} + \varepsilon_{u,i}$$

with $\varepsilon_{u,i}$ independent errors sampled from the same distribution centered at 0 and $\hat{\mu}$ the "true" rating for all movies.

Table 5: Just the Average

method	RMSE
Just the average	1.052046

4.4 Movie Effect Model

We know that movies are rated differently, some movies are just generally rated higher than others. This could happen due to popularity. As we see in Data Exploration section, the more often a movie is rated, the higher its average rating. Then we create a new model approach based in this formula:

$$Y_{u,i} = \hat{\mu} + b_i + \epsilon_{u,i}$$

The b_i is a measure for the popularity of movie i, i.e. the bias of movie i.

$$\hat{b}_i = mean\left(\hat{y}_{u,i} - \hat{\mu}\right)$$

Table 6: Movie Effect

method	RMSE
Just the average	1.0520464
Movie Effect Model	0.9410267

4.5 Movie + User Effect Model

We knows that people have different personality. In this case, some users loves every movies that they see, and some other are grumpier. Hence we can implement this user effect into our model. Now, the formula looks like this:

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

The b_u is a measure for the mildness of user u, i.e. the bias of user u.

$$\hat{b}_u = mean\left(\hat{y}_{u,i} - \hat{\mu} - \hat{b}_i\right)$$

Table 7: Movie + User Effect

method	RMSE
Just the average	1.0520464
Movie Effect Model	0.9410267
Movie + User Effect Model	0.8578161

4.6 Movie + User + Genres Effect Model

We know that users have different genres taste. Some users like drama, other like adventure or comedy. We represent this in this formula:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_{u,g} + \epsilon_{u,i}$$

The b_g is a measure for how much a user u likes the genre g.

$$\hat{b}_g = mean\left(\hat{y}_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u\right)$$

Table 8: Movie + User + Genre Effect

method	RMSE
Just the average	1.0520464
Movie Effect Model	0.9410267
Movie + User Effect Model	0.8578161
Movie + User + Genres Effect Model	0.8577266

4.7 Movie + User + Genres + Release Date Effect Model

The fourth bias to adjust for within the model was the release date of the movie. The exploratory analysis in the previous section showed an effect of the release year, b_y, and the least squares estimate of the date effect, $\hat{b}_r$ calculated using the formula shown below, building on the algorithm developed already.

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_g + b_r + \epsilon_{u,i}$$

$$\hat{b}_r = mean\left(\hat{y}_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u - \hat{b}_g\right)$$

Table 9: Movie + User + Genres + Release Date Effect

method	RMSE
Just the average	1.0520464
Movie Effect Model	0.9410267
Movie + User Effect Model	0.8578161
Movie + User + Genres Effect Model	0.8577266
Movie + User + Genres + Release Date Effect Model	0.8573800

4.8 Regularize Model

To improve our results, we will use **regularization**. Regularization constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes. To estimate the b's, we will now **minimize this equation**, which contains a penalty term:

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

The first term is the mean squared error and the second is a penalty term that gets larger when many b's are large.

The values of b that minimize this equation are given by:

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

where n_i is a number of ratings b for movie i.

The **larger** λ is, the more we shrink. λ is a tuning parameter, so we can use cross-validation to choose it. Here, the regularization model was developed to adjust for all of the effects previously described, as shown below. A range of values for λ (range: 0-10, with increments of 0.25) was applied in order to tune the model to minimize the RMSE value. As before, all tuning was completed within the edx dataset, using the train and test sets, so as to avoid over-training the model in the validation set.

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i} - \mu - b_i - b_u - b_g - b_r\right)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2 + \sum_r b_r^2\right)$$

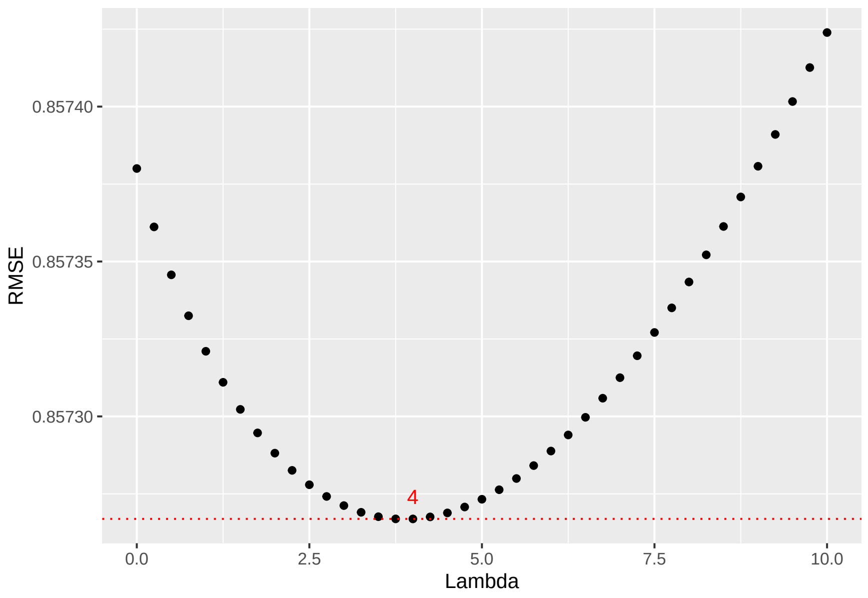

Figure 10: Optimal Lambda

Table 10: Regularized Movie + User + Genres + Release Date Effect

method	RMSE
Just the average	1.0520464
Movie Effect Model	0.9410267
Movie + User Effect Model	0.8578161
Movie + User + Genres Effect Model	0.8577266
Movie + User + Genres + Release Date Effect Model	0.8573800
Regularized Movie + User + Genres + Release Date Effect Model	0.8572669

4.9 Final Test with validation Data Set

Having refined the model algorithm within the train and test sets created by partitioning edx, the final stage of the project was to train the algorithm using the full edx dataset and then to predict ratings within the validation dataset (final_holdout_test). The final model, adjusting for biases introduced by movie, user, genre, release date, and collectively regularized using the optimal value for λ, was used to predict ratings in the validation dataset, and to calculate the final validation RMSE. And the final RMSE result:

```
## [1] 0.8634053
```

5 Conclusion

The target of this project was to develop a recommendation system using the MovieLens 10M dataset that predicted ratings with a residual mean square error of less than 0.86490. Adjusting for a number of estimated biases introduced by the movie, user, genre, release date, and then regularizing these in order to constrain the variability of effect sizes, met the initially proposed objective: reaching an RMSE 0.8634053.

Although the algorithm developed here met the project objective it still includes a substantial error loss, which suggests that there is still scope to improve the accuracy of the recommendation system with some other techniques. One such approach is matrix factorization, a powerful technique for user or item-based collaborative filtering based machine learning which can be used to quantify residuals within this error loss based on patterns observed between groups of movies or groups of users such that the residual error in predictions can be further reduced.

The techniques used in this project were limited due to the impracticality of using some powerful tools to train such a large dataset on a personal computer. Thus, further work on the recommendation system developed here would focus on the use of matrix factorization which approach is more computationally demanding.