HarvardX Capstone MovieLens Project

Raphael Oliveira Abreu

2023-02-05

Introduction

Over the last decade, data analysis and data science have experienced explosive growth and have become integral parts of many industries, including business, finance, healthcare, social sciences, and many others. Data science has also become a key driver of innovation and business growth, with companies using data-driven insights to make informed decisions and drive revenue. As the amount of data continues to grow, the field of data analysis and data science is expected to become even more important in the years to come.

Before gaining popularity it has today, in 2006, Netflix opened a competition called Netflix Prize for the best collaborative filtering algorithm to predict user rating based on films. This capstone project shares objectives with the Neflix Prize. The goal of this project is to explore, develop and train a machine learning model recommendation system based on the MovieLens 10M dataset, a data set containing 10 millions ratings applied for 10 thousand movies by 72 thousand users. The goal of this project is to predict ratings with a root mean square error (RMSE) of less than 0.86490.

First, it will be conducted an exploratory data analysis of the validation set utilizing data manipulation and common visualization approaches. Then, it will be developed, trained and tested a recommendation system algorithm. At the end, the results will be shared followed by recommendations to be taken into account in the future.

Analysis

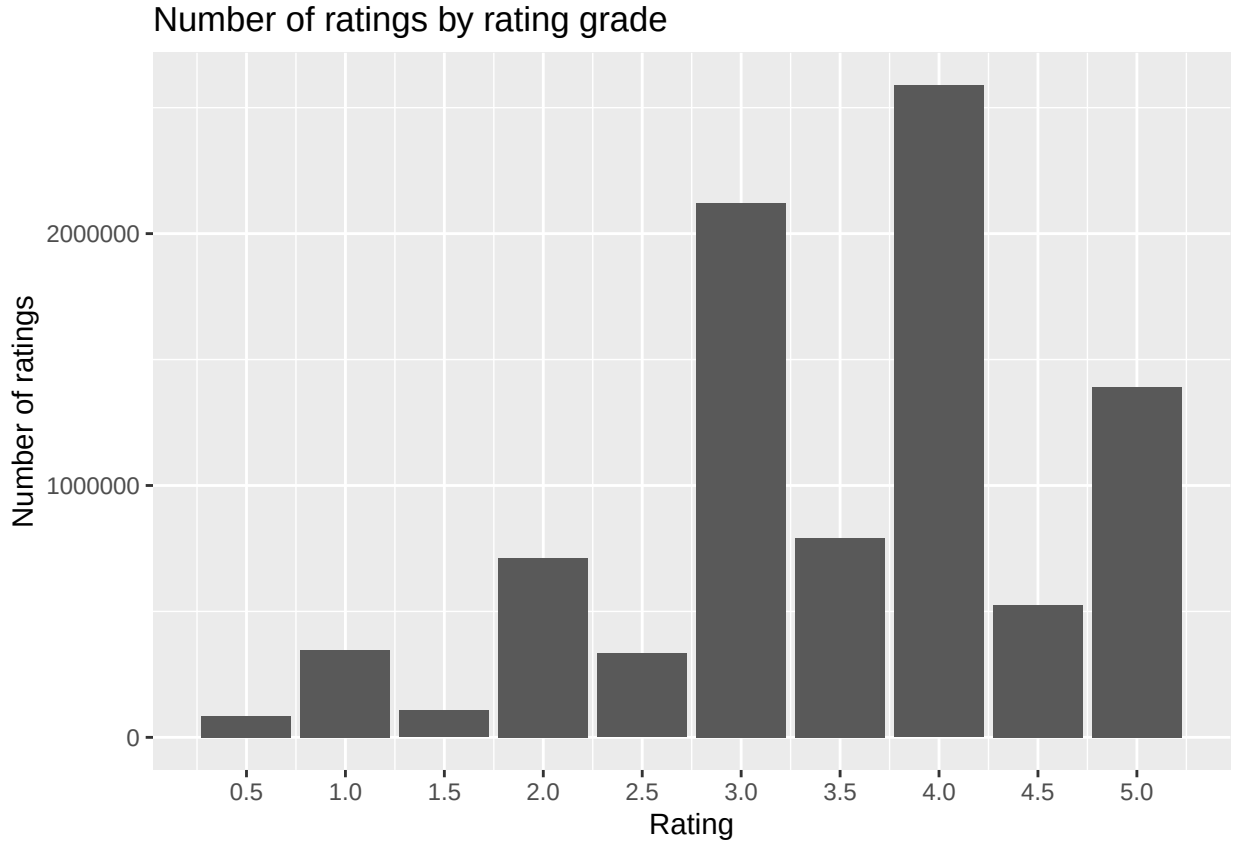
The sets edx and final_holdout_test used in this project where created based on code provided by the course instructions. Here are the results of the data exploration conducted on the edx data set.

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

The edx set has is a data.frame type object with 9,000,055 rows and 6 columns: userId, movieId, rating, timestamp, title and genres.

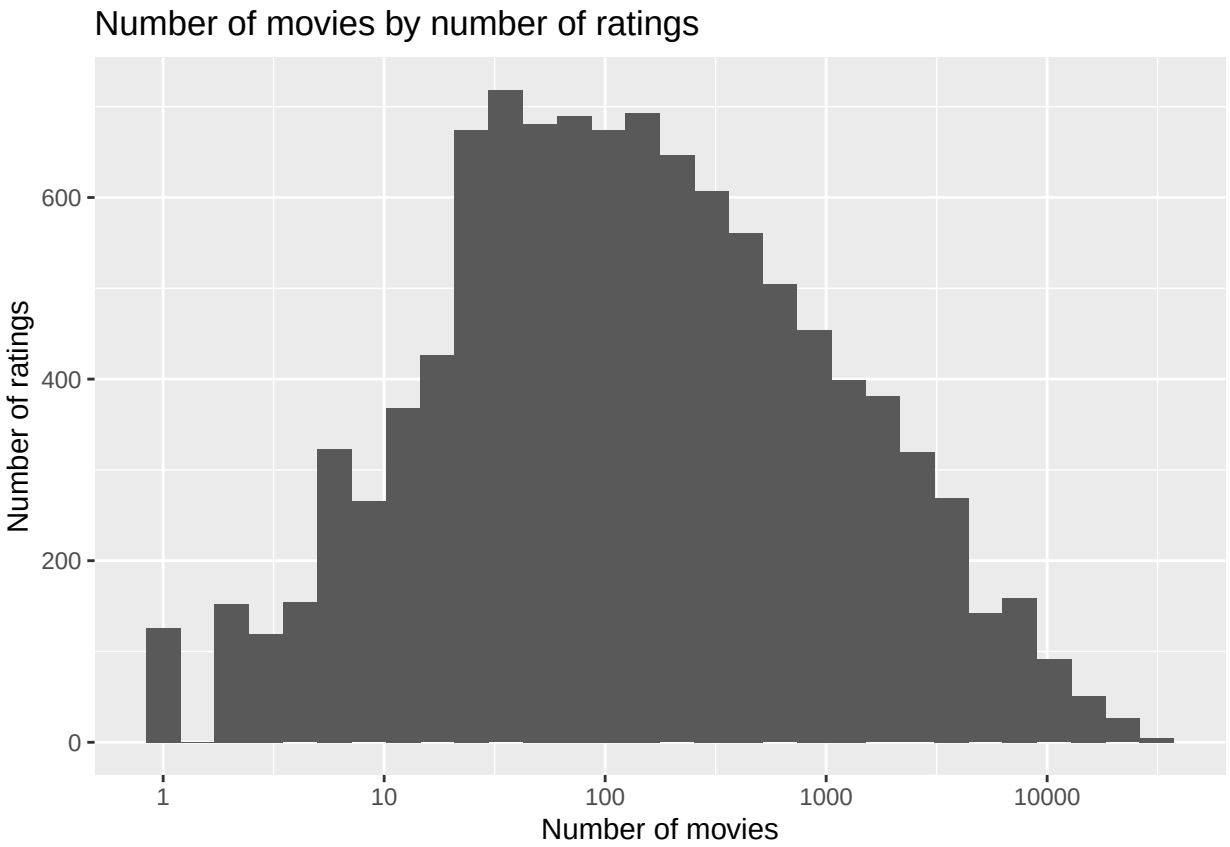
Ratings

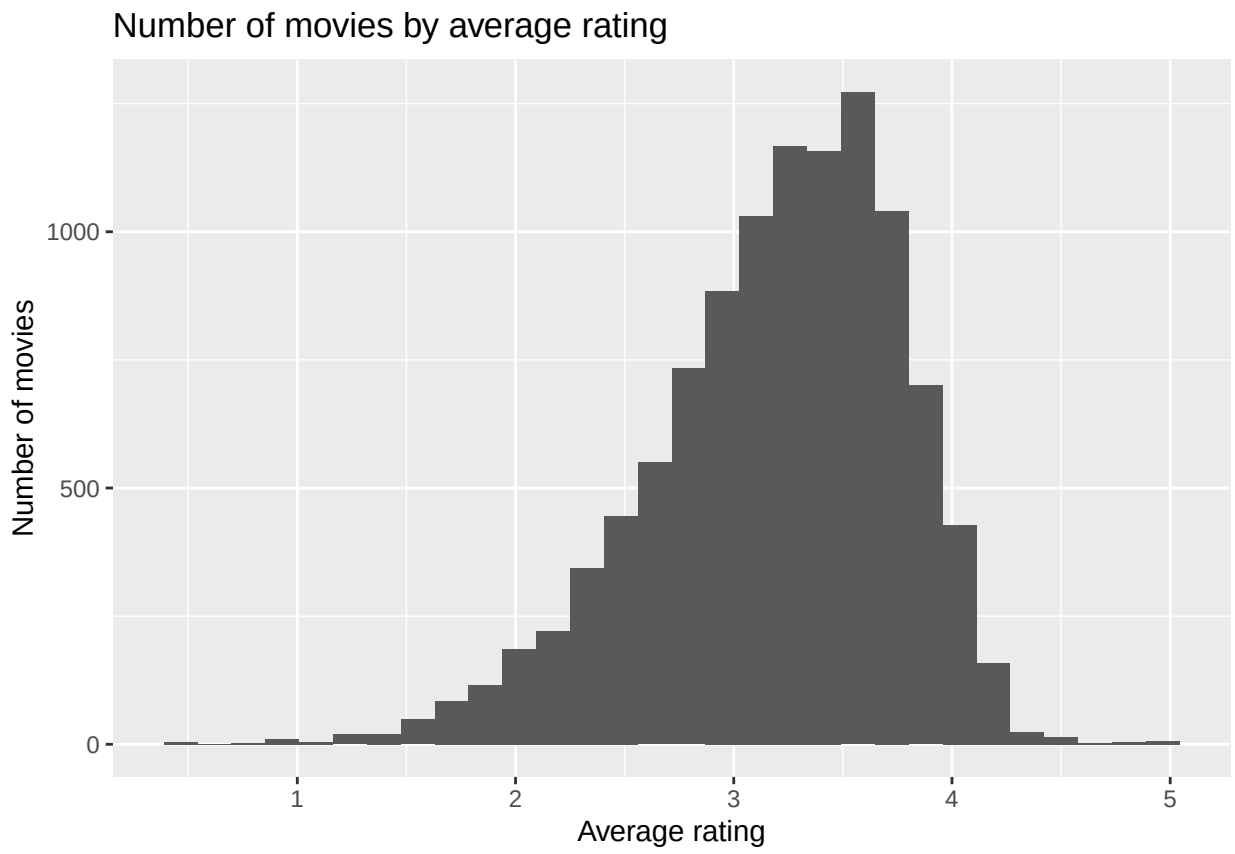
The edx set has a total of 9,000,055 ratings by users. These ratings have 10 possible values, ranging from 0.5 to 5 in increments of 0.5. The distribution of the ratings can be see in the figure below. Also worth noting that the average rating of the whole set is approximate 3.51.



Movies

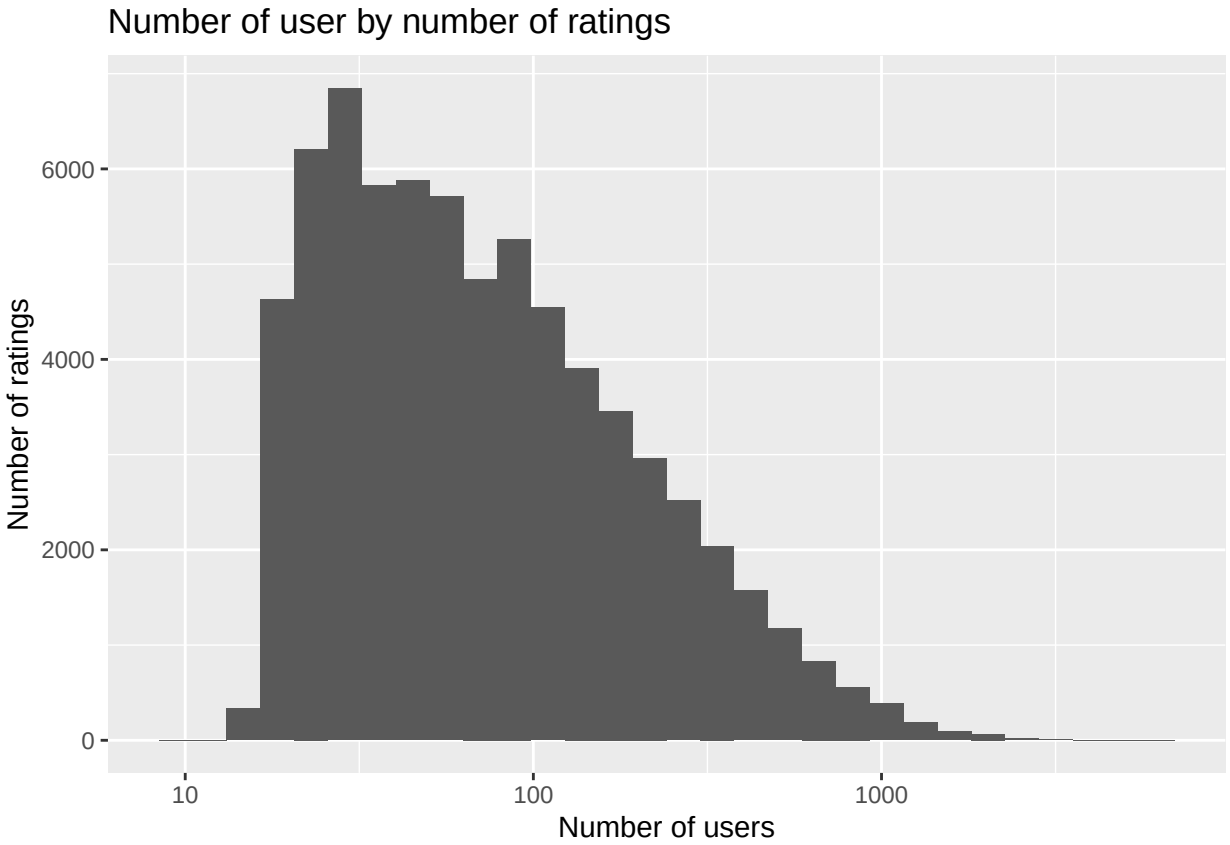
The set has 10677 unique movies which are cataloged with 20 genres classifications available. The movie with the most ratings is Pulp Fiction with 31362 ratings. The bottom of the list is composed of 126 movies receiving only one rating. When we make a distribution of the movies by average rating the shape is very similar to users by average rating.

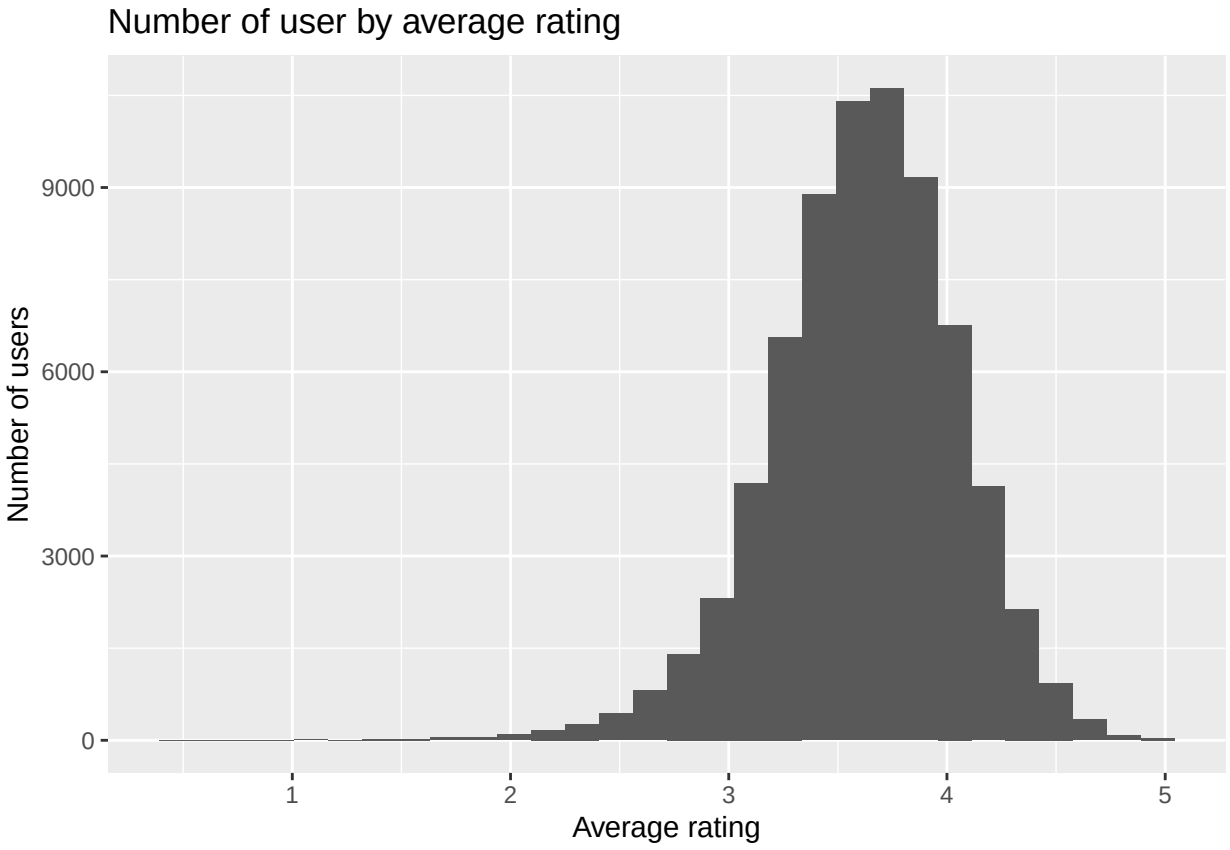

Number of movies by number of ratings
Number of ratings
600
400
200
0
1
10
100
1000
10000
Number of movies



Users

By exploring the userId variable the minimum number of movies a user reviewed is 10 times by one single user, and the maximum is 6616. 3470 users reviewed as few as 20 movies. It is also safely to say that no user reviewed the same movie twice since there is no duplication between the first and second variables, userId and movieId. Distributing the users of the dataset by the average rating each user have we can come upon a bell shaped figure.


Number of user by number of ratings
Number of ratings
6000
4000
2000
0
10
100
1000
Number of users



Genres

As said before, the data set utilizes 20 available genre types for the classification of the movies. Below, we see how many times each type was used for classification and the average that genre got across the full edx dataset.

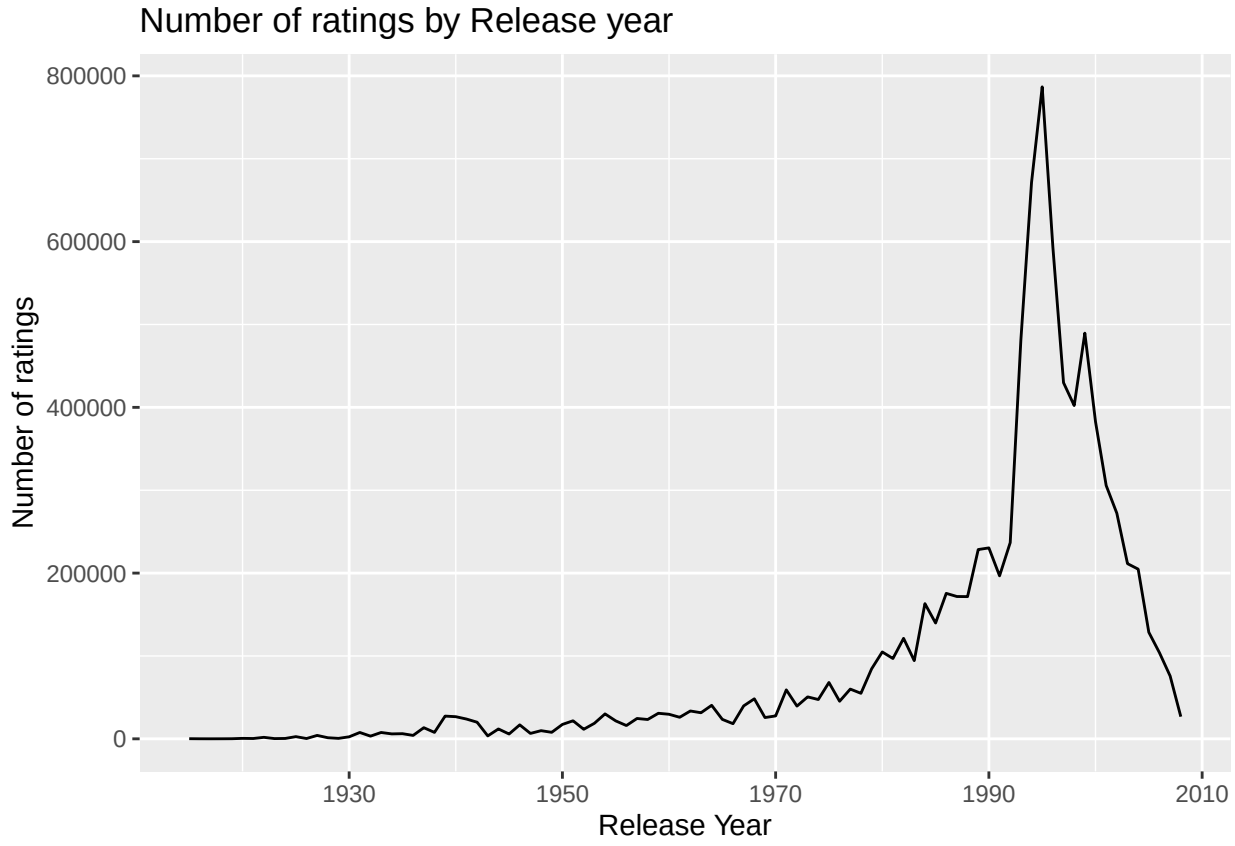
genres	count	rating
Drama	3910127	3.673
Comedy	3540930	3.437
Action	2560545	3.421
Thriller	2325899	3.508
Adventure	1908892	3.494
Romance	1712100	3.554
Sci-Fi	1341183	3.396
Crime	1327715	3.666
Fantasy	925637	3.502
Children	737994	3.419
Horror	691485	3.270
Mystery	568332	3.677
War	511147	3.781
Animation	467168	3.601
Musical	433080	3.563
Western	189394	3.556
Film-Noir	118541	4.012

genres	count	rating
Documentary	93066	3.783
IMAX	8181	3.768
(no genres listed)	7	3.643

Title (and release year)

The title variable contains the title of the movie with the addition of the year of release between parenthesis. A mutated dataset (edx_m) is build in order to extract and explore if the year of release has any influence in the ratings, and still keep the original edx dataset.

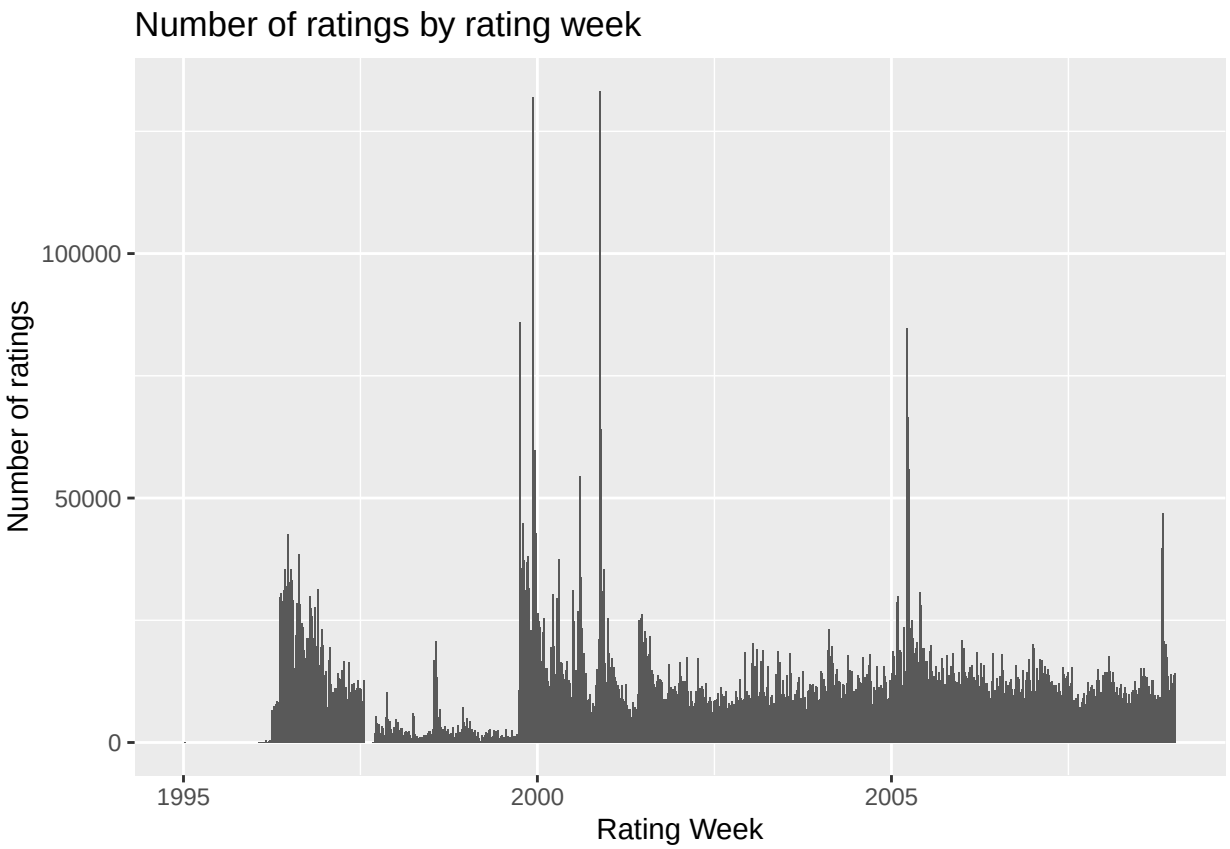
90's movies appears as the most reviewed release years followed by 2000's movies. This makes sense since these are the eras are contemporary to the in which the ratings where made. 80's movies follows in third followed by its preceding decade. This trend can be seen thought the rest of the release years.



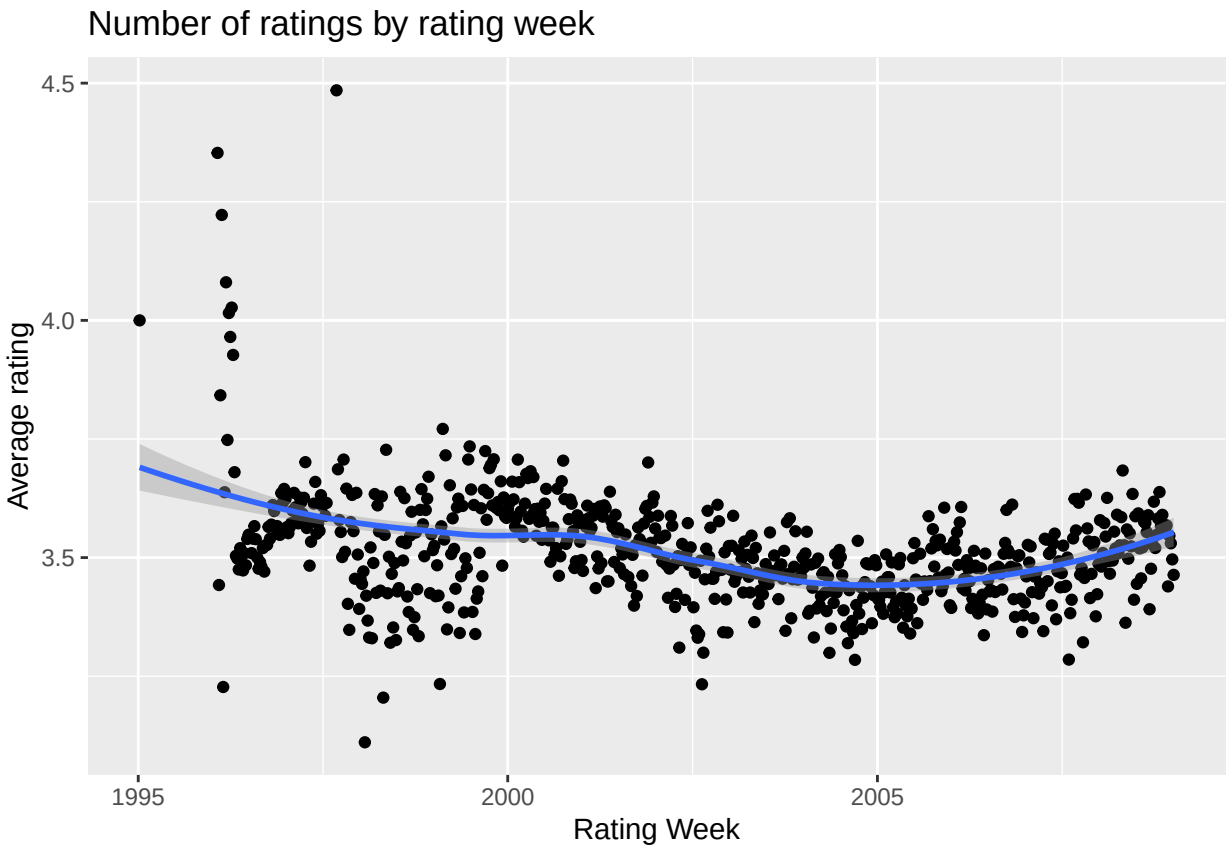
Timestamp

The timestamp column records the time the rating review was made, but it records in an specific way. The timestamp is the number of seconds that have elapsed since midnight January 1, 1970. So before continuing the analysis, it was necessary to transform the variable into a date format. The timestamp was also mutated into two new columns: one agglutinating dates into weeks, since the visualization of day to day data would be extremely crowded, and the second regarding the four seasons, which was made with the assistance of the hydroTSM package.

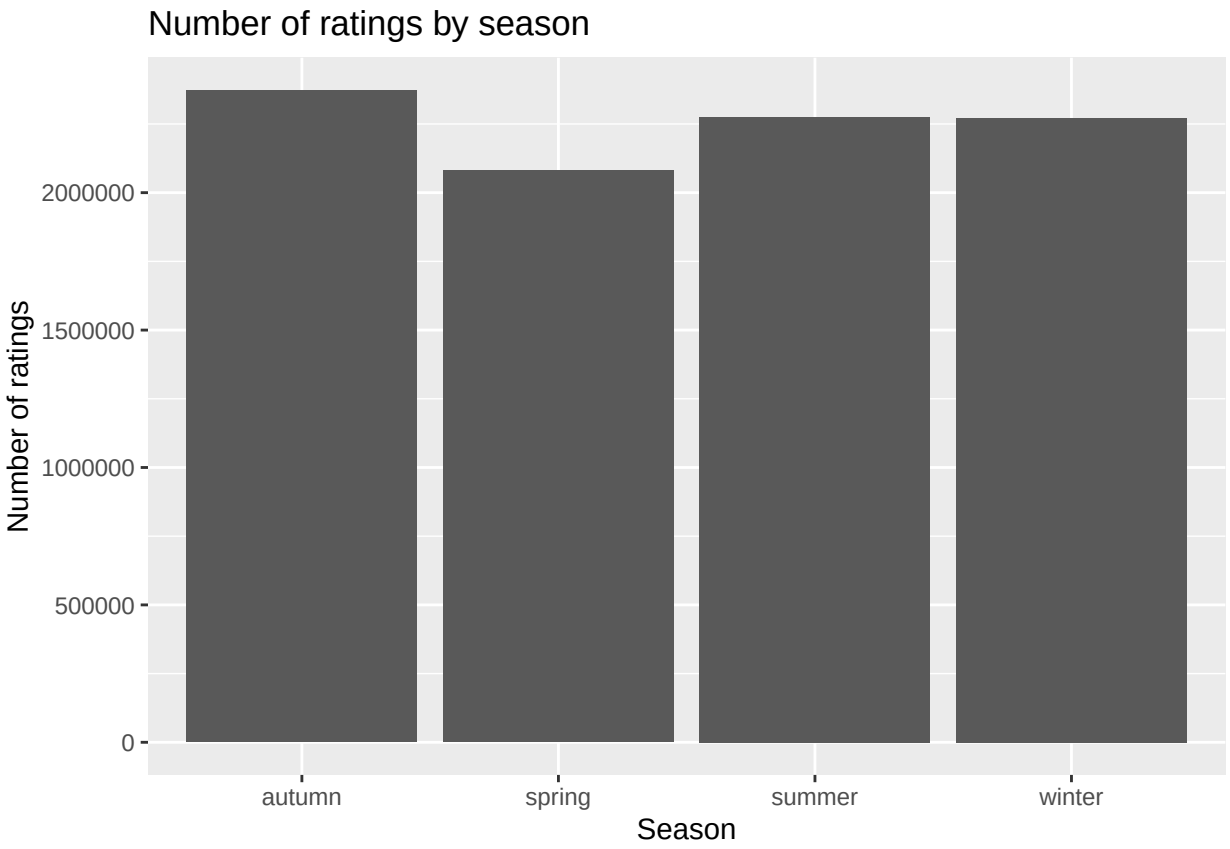
The dataset rating dates spams though almost 15 year, from 1995 to 2009. 1995 is the year with the least reviews, only 2. While 2000 is the year with the most reviews.

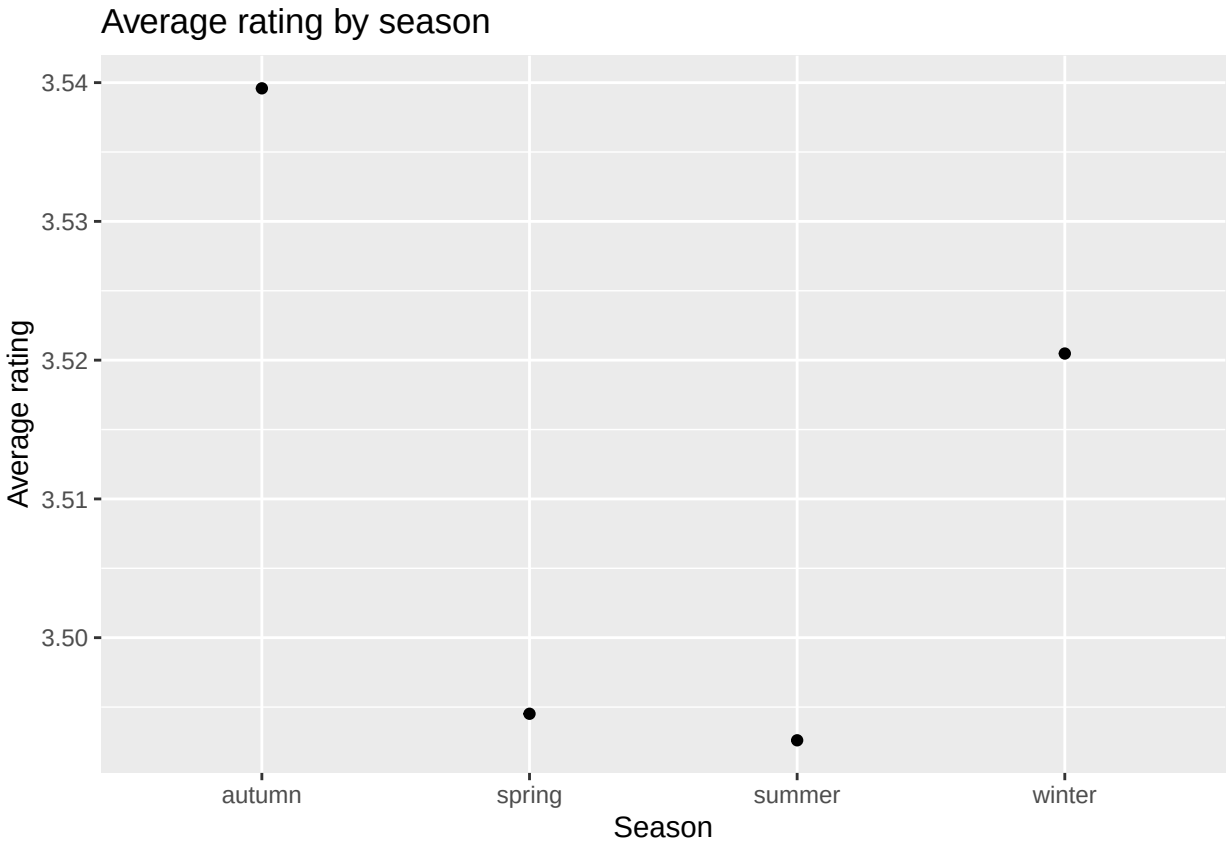


At first, there is a downward trend in average ranting throughout the first years that becomes stable from 1998 forward. The downward trend is more accentuated when dates are grouped by year.



Analysing timestamp as seasons of the year no trend can be seen, since all seasons have approximate the same number of reviews, although autumn has slightly more ratings than the average and spring is the season with the least number of reviews. When analyzing the average rating of each season no significant differences can be found between the seasons. All season rating are in a 0.05 interval.


Number of ratings by season
Number of ratings
2000000
1500000
1000000
500000
0
autumn
spring
summer
winter
Season



Method and Results

RMSE

The RMSE measures the average distance between the predicted ratings and the actual ratings in the test set. We can interpret it as the typical error we make when predicting a movie rating. It is a commonly used metric to evaluate the performance of regression models, including those used for predicting movie ratings. The lower the RMSE, the better the model is at predicting ratings.

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

where:

- N is the total number of user/movie combinations.
- $\hat{y}_{u,i}$ is the predicted rating for the same movie and user.
- $y_{u,i}$ is the actual rating provided by user u for movie u.

The project aimed to create an algorithm that achieved an RMSE value below 0.86490, as specified in the project objectives.

Algorith

Before any calculations the splitting of the edx data set must occur in order to save the final_holdout set and still have a train set and a test set. The set will be splitted with the proportions 80% for the train set and 20% for the test set.

Naive RMSE

The simplest prediction model is predicting the same rating for all movies regardless of user u and movie i meaning that the final result being the same is simply a random coincidence. This model can the written like this:

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

Where:

- $Y_{u,i}$ is the actual rating for movie i by user u
- μ represents the sum of all the movies i and users u
- $\epsilon_{u,i}$ is the independent erros sampled from the same distribution centered at zero.

The average of all ratings is the estimate of μ that minimizes the RMSE. Thus, $\hat{mu}$ as the mean of the ratings of the train set is the simple formula used to train the first algorithm.

```
## Warning: ‘data_frame()‘ was deprecated in tibble 1.1.0.
## i Please use ‘tibble()‘ instead.
```

method	RMSE
Project objective	0.86490
Simple Average	1.05984

With a RMSE higher than 1 we can say that the predicted ratings is 1 star away from the actual rating.

Adding the movie factor

The previous section revealed that ratings were not evenly distributed among all movies. Some movies received higher average ratings than others, and taking this bias into account would enhance the prediction accuracy. Here, this effect can be translated into b_i.

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

method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672

Adding the user factor

The previous section also revealed that ratings were not evenly distributed among all users and the majority ranging in middle. Some could also have given a bad rating for a good movie and vice-versa. Consequently, the training algorithm can be fine-tuned with the addition of the factor b_u for users into the model, which behavior is similar to b_i for movies. The model can be improved once again.

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

method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672
Movie + User Effect Model	0.8652

Adding the genre factor

The variable of movie genre also was one of the factors in which the rating depended also had an effect on the average rating given to said movie and user. Here, b_g will represent this effect.

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

method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672
Movie + User Effect Model	0.8652
Movie + User + Genre Effect Model	0.86484

Adding the release year factor

The date of review also had an effect on the average rating given to said movie and user. As demonstrated in the exploratory section, movies of a certain decade and years had on average. Here, b_y represents this effect.

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

method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672
Movie + User Effect Model	0.8652
Movie + User + Genre Effect Model	0.86484
Movie + User + Genre + Release Year Effect Model	0.86466

Adding the rating date factor

The fifth and last bias for adjusting the model is the rating date of the movie i by user u. The date of review also had an effect on the average rating given to said movie and user. To adjust for the swift changes in rating by date, the rating week will by used to smooth these abrupt changes. Here, b_r will represent this effect.

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

method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672
Movie + User Effect Model	0.8652
Movie + User + Genre Effect Model	0.86484
Movie + User + Genre + Release Year Effect Model	0.86466
Movie + User + Genre + Release Year + Rating Date Effect Model	0.86451

Regularizing the model

Regularization, simply put, permits us to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for the large values of b to the sum of squares equations that we minimize. Since having many large bs make it harder to minimize the equation that we're trying to minimize. This is done with the addition of a penalty term that gets larger when many bs are large to the residual sum of square.

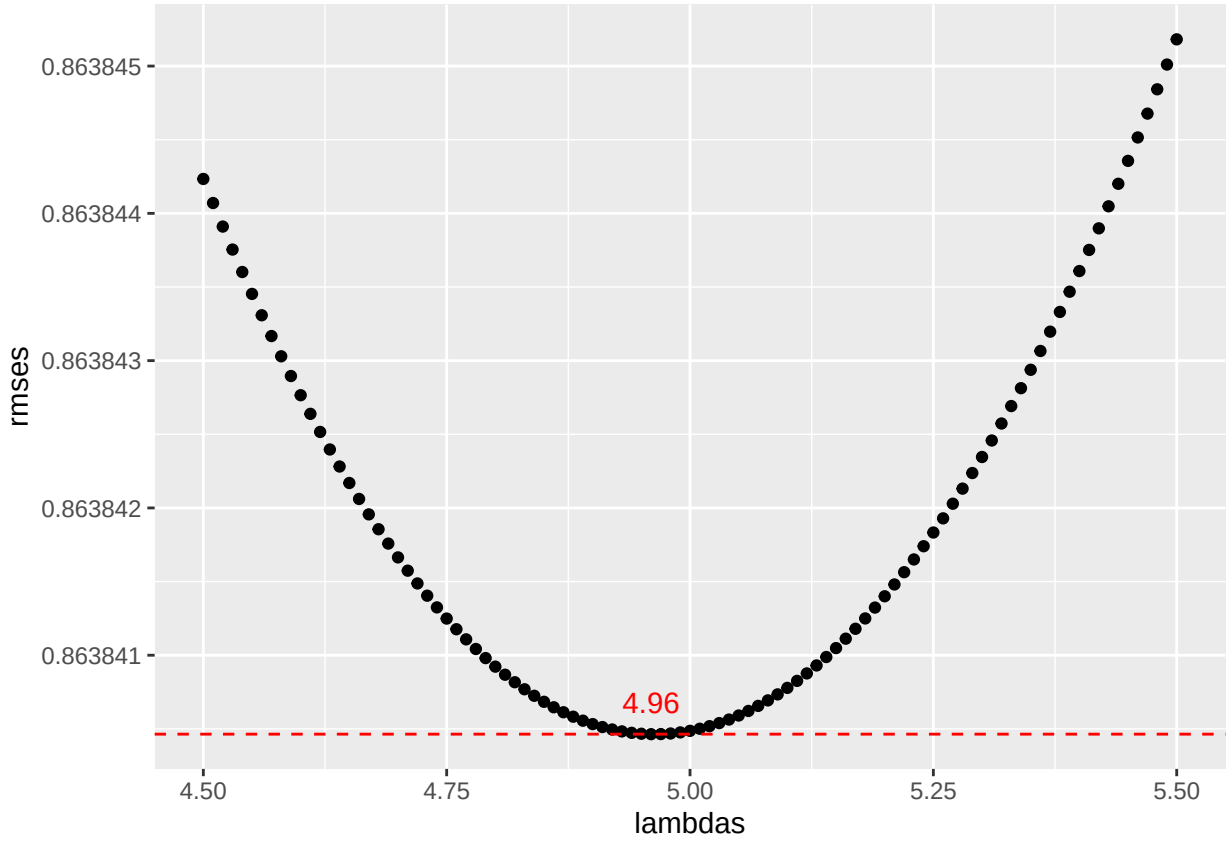
$$\frac{1}{N}\sum_{u,i}\left(y_{u,i} - \mu - b_i - b_u - b_g - b_y - b_r\right)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2 + \sum_y b_y^2 + \sum_r b_r^2\right)$$

This function can be rewritten like this:

$$\hat{b}_r\left(\lambda\right) = \frac{1}{\lambda + n_r}\sum_{u=1}^{n_r}\left(Y_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u} - \hat{b_g} - \hat{b_y}\right)$$

When n_r is very large which will give us a stable estimate, then lambda is effectively ignored because n_r plus *lambda* is about equal to n_r. However, when n_i is small, then the estimate of b_r is shrunken towards zero. The larger *lambda* the more we shrink.

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



method	RMSE
Project objective	0.86490
Simple Average	1.05984
Movie Effect Model	3.63672
Movie + User Effect Model	0.8652
Movie + User + Genre Effect Model	0.86484
Movie + User + Genre + Release Year Effect Model	0.86466
Movie + User + Genre + Release Year + Rating Date Effect Model	0.86451
Reguralized User + Movie + Genre + Release year + Rating Date Effect Model	0.86384

Testing out the final model

Finally, here is result of the final model on the movielens final_holdout_test set.

method	RMSE
Project objective	0.86490
Final Model Validation	0.86405

Conclusion

The aim of this project was to create a recommendation system that could predict a RMSP of less than 0.86490 using the MovieLens dataset. To achieve it, the approach involved adjusting for biases caused by

movie, user, genre, release year and review date (based on a week round). The next step in the approach was regularizing these biases to constrain the variability of effect sizes. The final outcome in the test set was a RMSE of 0.86384. When the algorithm was then implemented into the final_holdout_test set a RMSE of 0.86405 was produced.

References

- http://rafalab.dfci.harvard.edu/dsbook/large-datasets.html
- https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a