HarvardX Data Science Program

Movielens project

Do Quang Anh

2022-04-01

Contents

1 Overview

This project is related to the MovieLens Project of the HervardX: PH125.9x Data Science: Capstone course. The present report start with a general idea of the project and by representing its objectif.

Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict movie ratings until a final model. Results will be explained. Finally the report ends with some concluding remarks.

1.1 Introduction

Recommendation systems use ratings that users have given to items to make specific recommendations. Companies that sell many products to many customers and permit these customers to rate their products, like Amazon, are able to collect massive datasets that can be used to predict what rating a particular user will give to a specific item. Items for which a high rating is predicted for a given user are then recommended to that user.

The same could be done for other items, as movies for instance in our case. Recommendation systems are one of the most used models in machine learning algorithms. In fact the success of Netflix is said to be based on its strong recommendation system. The Netflix prize (open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films), in fact, represent the high importance of algorithm for products recommendation system.

For this project we will focus on create a movie recommendation system using the 10M version of MovieLens dataset, collected by GroupLens Research.

1.2 Aim of the project

The aim in this project is to train a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the inputs of a provided subset (edx dataset provided by the staff) to predict movie ratings in a provided validation set.

The value used to evaluate algorithm performance is the Root Mean Square Error, or RMSE. RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, a lower RMSE is better than a higher one. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers. Four models that will be developed will be compared using their resulting RMSE in order to assess their quality. The evaluation criteria for this algorithm is a RMSE expected to be lower than 0.8775. The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

Finally, the best resulting model will be used to predict the movie ratings.

1.3 Dataset

The MovieLens dataset is automatically downloaded

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

2 Methods/Analysis

2.1 Setting the Data

First the data is imported and partitioned. Normally, the code below would be used.

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                            title = as.character(title),
                                            genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
saveRDS(edx, file = "edx.rds")
saveRDS(validation, file = "validation.rds")
```

Dataset	Number of Rows	Number of Columns
edx	9,000,055	6
validation	999,999	6

	x
userId	FALSE
movieId	FALSE
rating	FALSE
timestamp	FALSE
title	FALSE
genres	FALSE

```r
#Load the Data
edx <- readRDS("edx.rds", refhook = NULL)
validation <- readRDS("validation.rds", refhook = NULL)
```

2.1.1 Dataset Dimenions

Our training set is edx and test set is validation. Together, there are 10,000,054 records

After generating codes provided in the project overview, we can see that the edX dataset is made of 6 features for a total of about 9,000,055 observations.The validation set which represents 10% of the 10M Movielens dataset contains the same features , but with a total of 999,999 occurences. we made sure that userId and movieId in edx set are also in validation set.

Each row represents a rating given by one user to one movie. The column "rating" is the outcome we want to predict, y. Taking into account both datasets, here are the features and their characteristics:

quantitative features

-userId : discrete, Unique ID for the user.

-movieId: discrete, Unique ID for the movie.

-timestamp : discrete , Date and time the rating was given.

qualitative features

-title: nominal , movie title (not unique)

-genres: nominal, genres associated with the movie.

outcome,y

-rating : continuous, a rating between 0 and 5 for the movie.

Check missing value in edx. There are no missing values in any column.

```r
sapply(edx, {function(x) any(is.na(x))})%>% niceKable
```

A preview of the data structure is shown below from the first few rows in 'edx'.

Rating is the dependent/target variable - the value we are tring to predict.

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

	x
userId	integer
movieId	numeric
rating	numeric
timestamp	integer
title	character
genres	character

2.1.2 Data Classes

We can see that userId, movieId and rating are not factors, despite each having a smaller number of unique values. Furthermore, timestamp (the timestamp of the rating) is not useful as an integer.

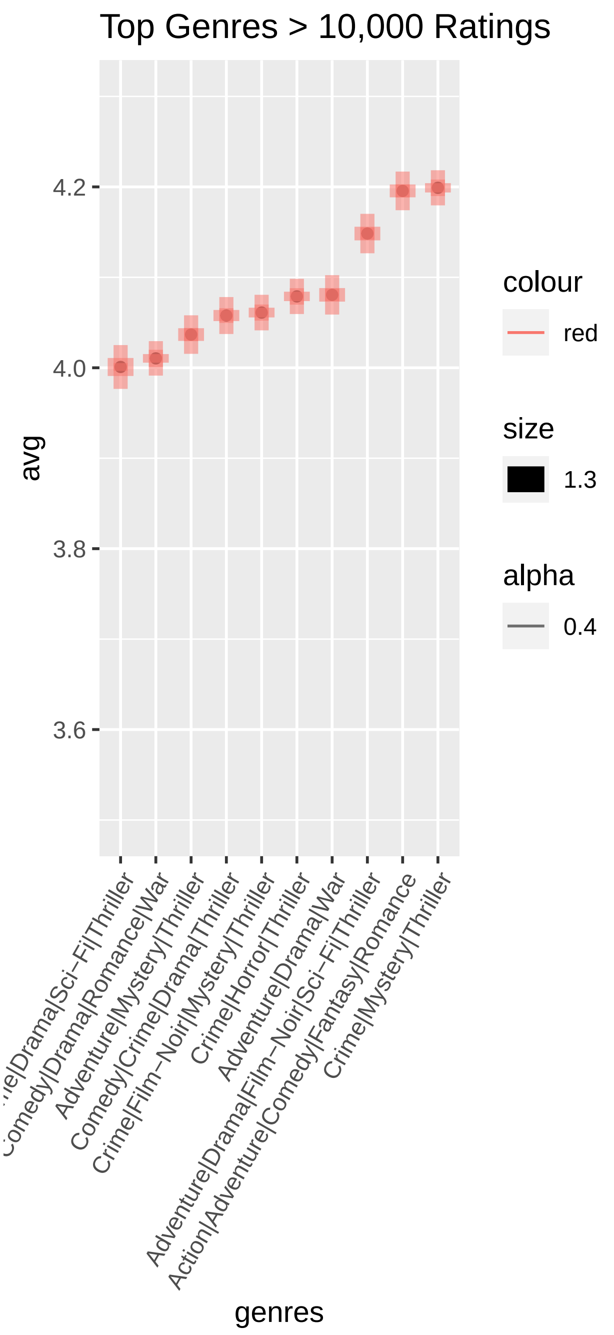
2.1.3 Genres

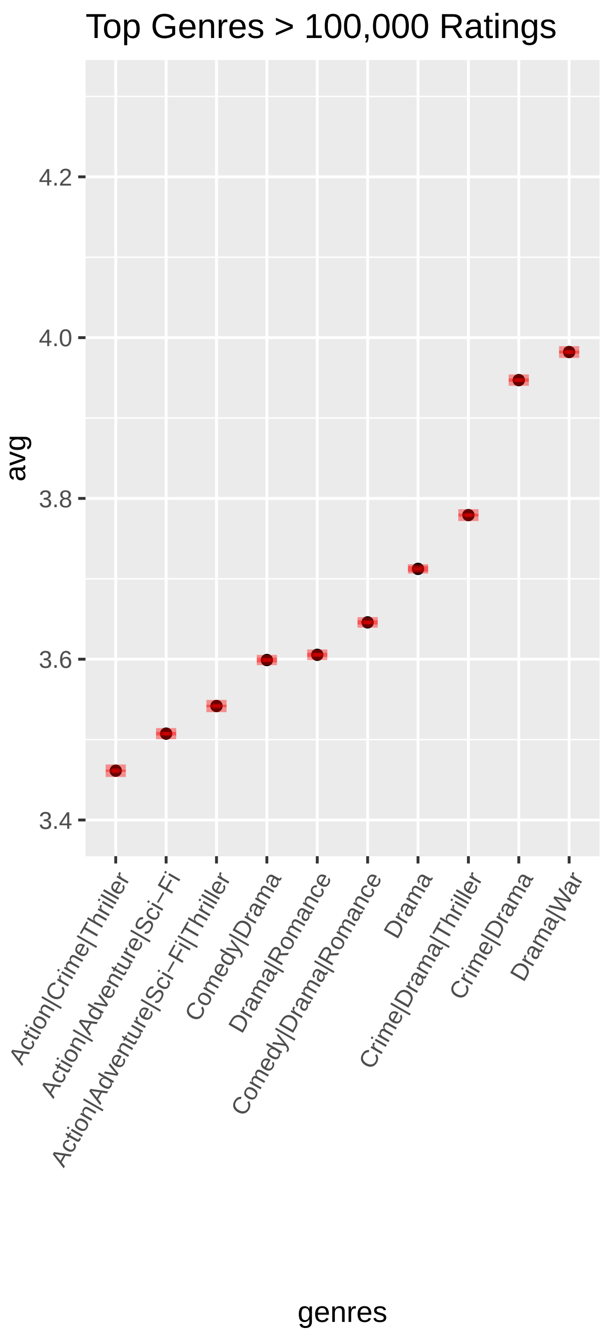
While there are only 20 unique genres, a film may be classified into multiple genres, up to seven at once. There are 797 of these unique combinations. Here are some of the biggest combinations.

2.1.4 Best-Rated Genres

Which genre combinations have the best average ratings? We'll look only at the top 10 genres from all genres that have over **10,0000** ratings. Each genre will have an error bar with the standard error of the rating.

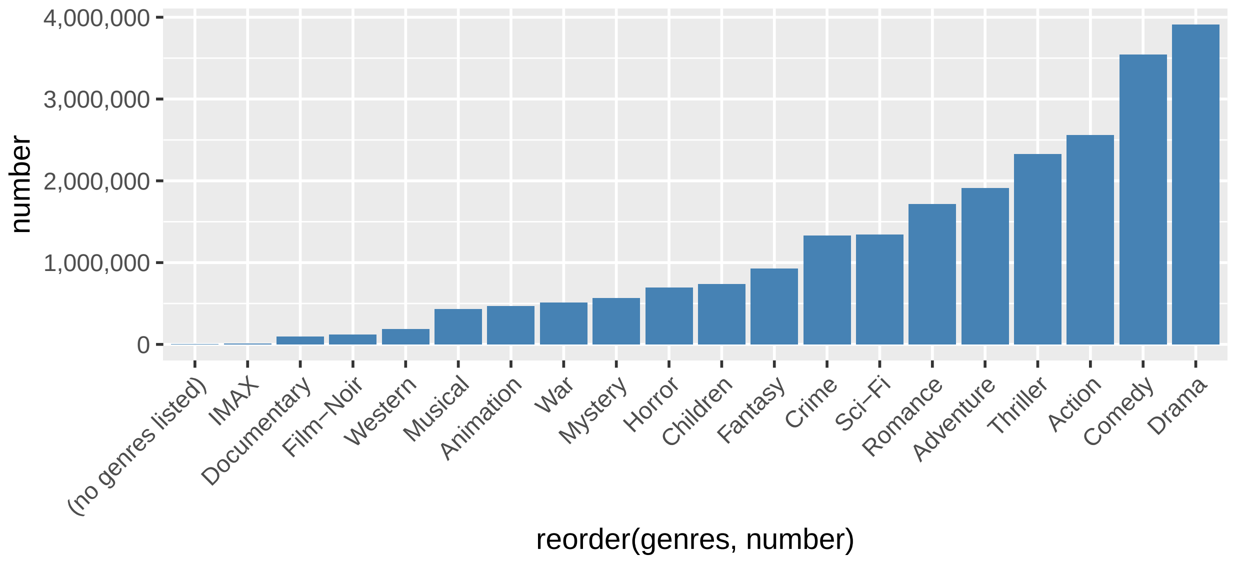
genres	genreCount
Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller	7
Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery	6
Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery	6
Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance	6





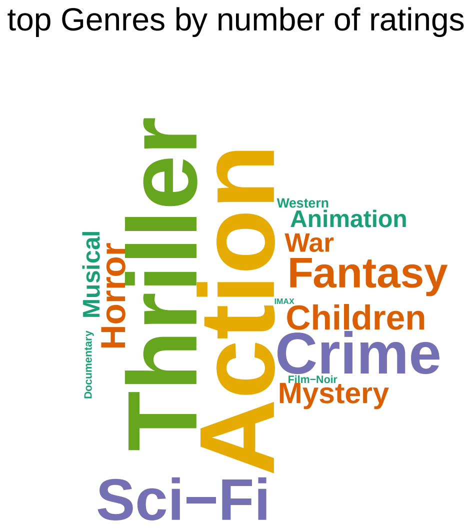
2.1.5 Genre prevalence

We may later decide to split these genre combinations up for better predictive value. Let's look at the individual prevalence of ratings each genre.



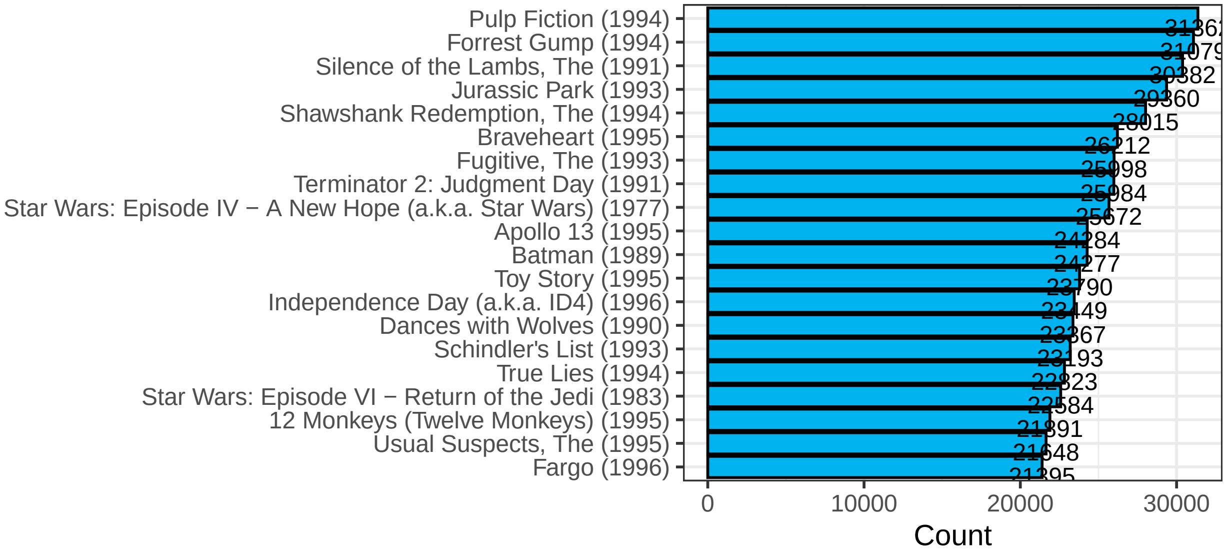
As expected, we can see some users are prolific, rating many movies. Likewise, some movies are very commonly rated.

```
library("tm")
layout(matrix(c(1,2), nrow =2) , heights = c(1,4))
par(mar=rep(0,4))
plot.new()
text(x=0.5,y=0.5, "top Genres by number of ratings")
wordcloud(words=genre_count$genres,freq=genre_count$number,min.freq=50,
     max.words = 20,random.order=FALSE,random.color=FALSE,
     rot.per=0.35,colors = brewer.pal(8,"Dark2"),scale=c(5,.2),
     family="Helvetica",font=2,
     main = "Top genres by number of ratings")
```



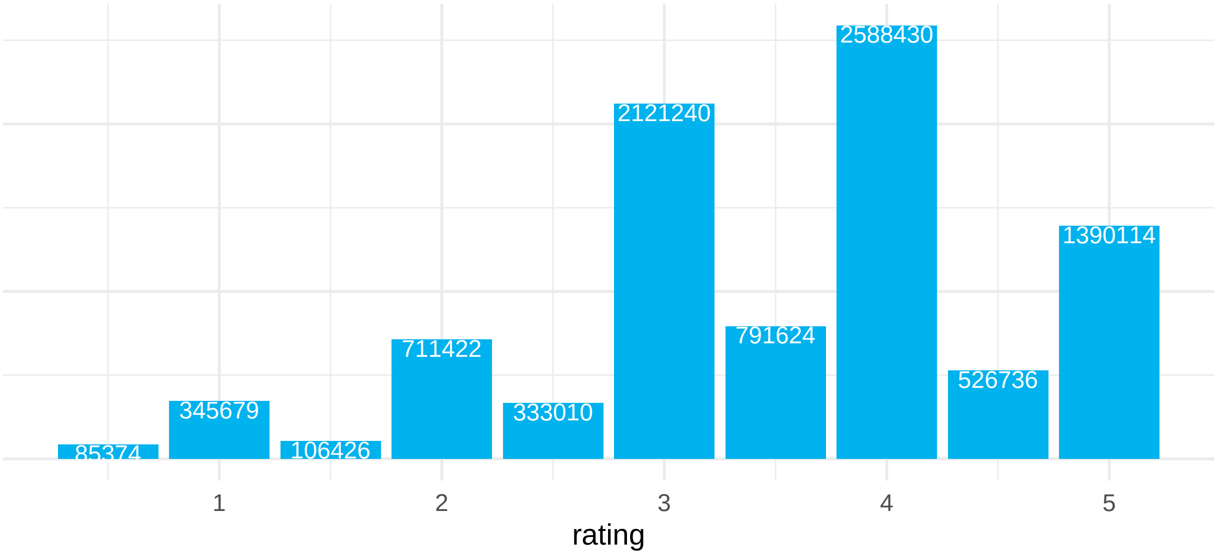
2.1.6 Reviews

Which films have the most popular(most rating)?

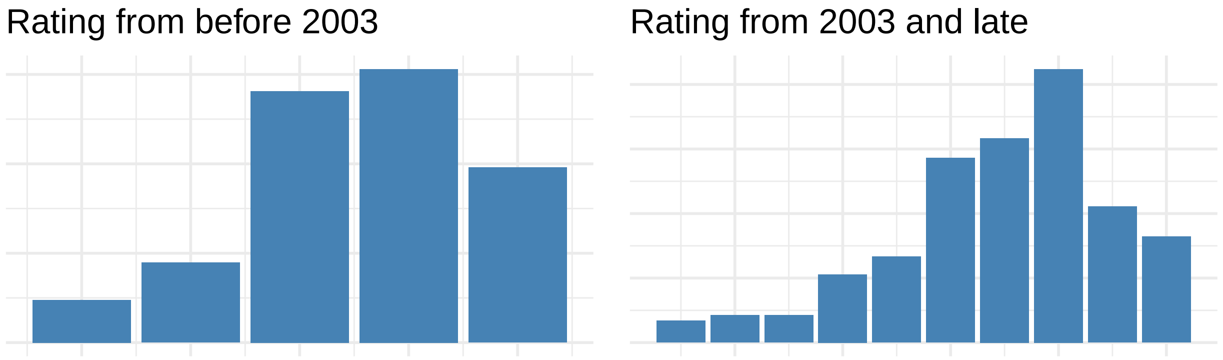


2.1.7 Rating Frequency

Half-star ratings are given out less frequently than full-stars. 4-star and 3-star ratings are the most common, followed by 5-star ratings.



Part of the reason for fewer half-star ratings is likely that half-stars weren't used in the *Movielens* rating system prior to 2003. Here are the distributions of the ratings before and after half stars were introduced.

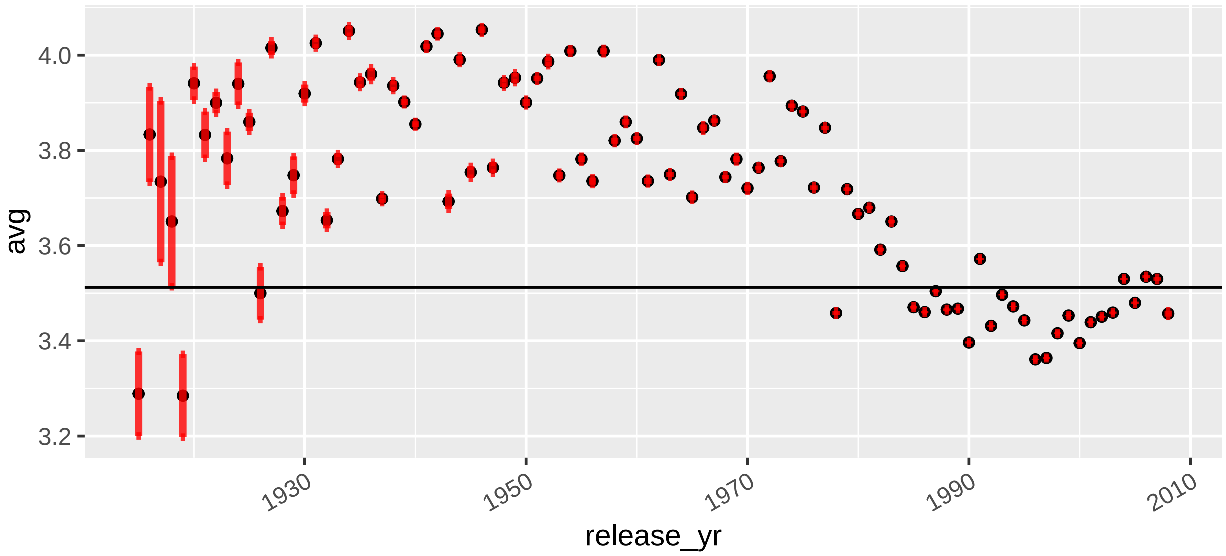


Rating from before 2003

Rating from 2003 and late

2.1.8 Average Rating By Year of Release

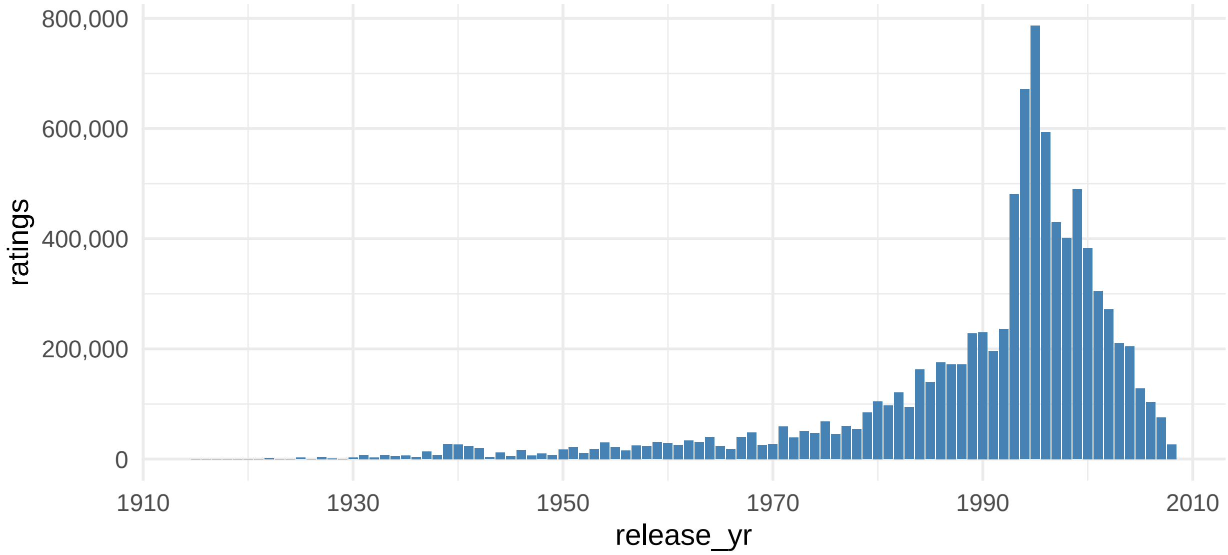
Earlier films have broader variability in their ratings, and films over 30 years old tend to be rated more highly on average. (The *Golden Age of Hollywood* is commonly held to have spanned from the late 1920s to the early 1960s).



Note: The chart above features standard error bars, and includes a reference line for the overall mean of all ratings More modern films have a tighter variability, and are generally located closer to the overall mean of 3.51. This is almost certainly due to the fact that older films have fewer ratings, and modern films have many

First_Quart	Mean_Rating	Median_Rating	Third_Quart
3	3.512465	4	4

Method	RMSE
Mean Rating (Naive)	1.061202



more.
### Summary Statistics

3 Results

3.1 Naive Prediction

We will make our first prediction, by simply using the mean of all ratings (3.51) as the estimate for every unknown rating. This will be the baseline against which model improvements will be evaluated.

3.2 Feature Engineering

Based on our first result, we will try to enhance the utility of the columns we have. It would be nice to split out genres into unique columns with a boolean 0/1 or TRUE/FALSE if a film exists in that genre (i.e *one-hot encoding*). This would help transform `genres` from a categorical variable to one that we can use for principal component analysis. (More on that in the Conclusion section.)

For now we will remove `genres` altogether. Then we will factorize userId and movieId, extract the movie's year of release from the title as `release_yr`, and calculate the number of years between a movie's release and a user's review as `review_dly`.

3.3 Bias Terms + Effects

3.4 Predict ratings based on movie effect

A bias term will be calculated for each movie to determine how much better/worse a given film is from the overall average, based on the average of all ratings for that film only. We will subtract the $overallMean$ ('r overall_mean') from the $movieMean$ for each film to determine the bias term, as below:

userId	movieId	rating	release_yr	review_dly
1	122	5	1992	4
1	185	5	1995	1
1	292	5	1995	1
1	316	5	1994	2
1	329	5	1994	2
1	355	5	1994	2

Method	RMSE
Mean Rating (Naive)	1.0612018
Movie Effect Model	0.9439087

$$Bias_{movie} = Mean_{movie} - Mean_{overall}$$

More positively rated films will have a positive bias value, while negatively rated films will have a neagative bias value. Does this improve the model?

When tested against unseen data, it's an improvement over the naive prediction.

3.5 Predict ratings based on user + movie effect

It is understood that users may have a tendency to rate movies higher or lower than the overall mean. Let's add this into the model. First we'll calculate the bias for each user:

$$Bias_{user} = Mean_{user} - Mean_{overall}$$

Then we'll combine the bias of a user, with the bias of a film and add both to the overall mean (3.51) for a combined bias rating for each unique combination of a user rating for a given film.

$UserMovieBias = overallMean + movieBias + userBias$

Let's check this against the test set of data.

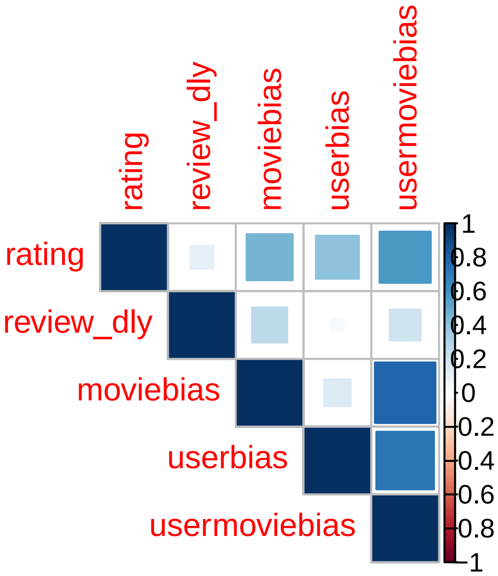
It's another improvement.

userId	movieId	rating	release_yr	review_dly	moviebias	userbias	usermoviebias
1	122	5	1992	4	-0.6538793	1.487535	4.346121
1	185	5	1995	1	-0.3831312	1.487535	4.616869
1	292	5	1995	1	-0.0944545	1.487535	4.905545
1	316	5	1994	2	-0.1627882	1.487535	4.837212
1	329	5	1994	2	-0.1750082	1.487535	4.824992
1	355	5	1994	2	-1.0246780	1.487535	3.975322

Method	RMSE
Mean Rating (Naive)	1.0612018
Movie Effect Model	0.9439087
User + Movie Effect Model	0.8850398

3.5.1 Correlation Plot

The improvement makes sense as we can see that our bias figures are closely and positively related with rating. The combibed user + movie bias term is most the value most positively correlated with rating.



3.6 Regularization

We know that films with very few ratings may be overinflated or underinflated. Their relative obscurity will cause very few ratings to determine the size of the movie effect. Likewise for users. Let's penalize films and users with very few ratings, by holding them closer to the $overallMean$ until the sample size n (the number of ratings by a specific user, or for a specific movie) increases.

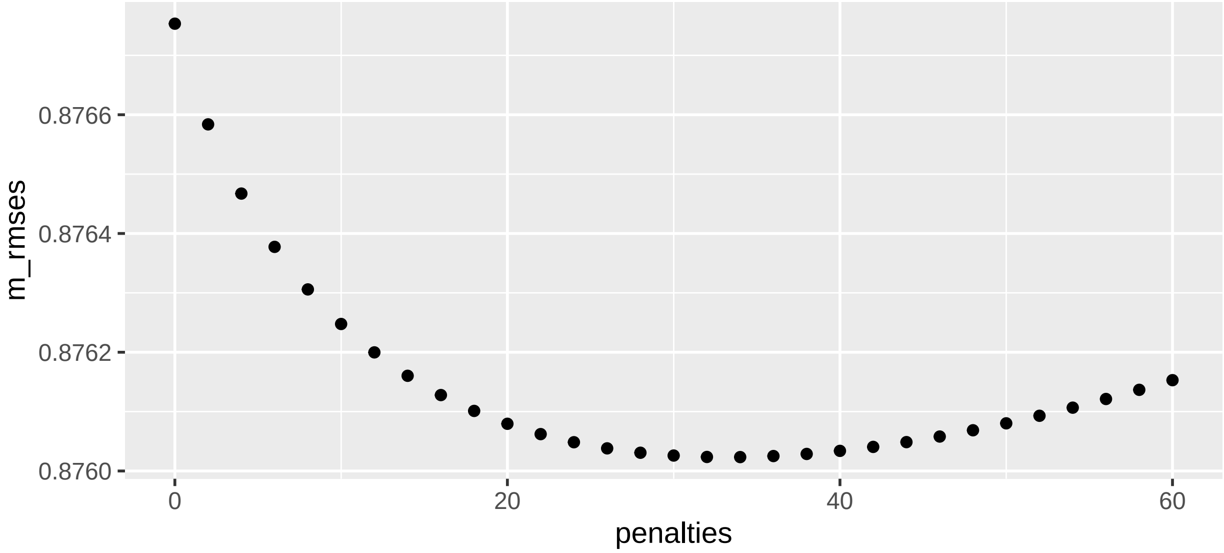
$$Bias_{user} = \frac{\sum(Mean_{ratingsbyuser} - Mean_{overall})}{n_{ratingsbyuser} + Penalty_{user}}$$

To do so, we will cross-validate a penalty figure within the *training* dataset. We will avoid cross-validating against the test set, as that would be cheating!

In setting the penalty, we will try figures from 0 to 60, increasing by increments of 2.

```
penalties <- seq(0, 60, 2)
```

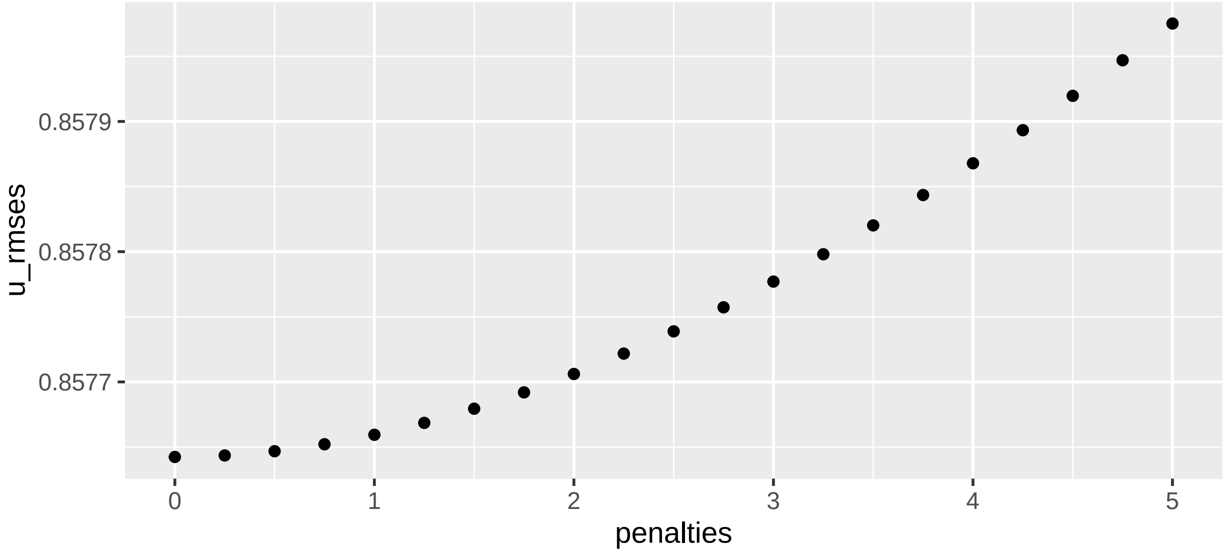
Method	RMSE
Mean Rating (Naive)	1.0612018
Movie Effect Model	0.9439087
User + Movie Effect Model	0.8850398
User + Regularized Movie Effects	0.8839696



The optimum value for our $Bias_{movie}$ figure is 34. How does the model improve when checked against the test set?

A little bit! Now let's do the same for user penalties.

```r
penalties <- seq(0, 5, 0.25)
```

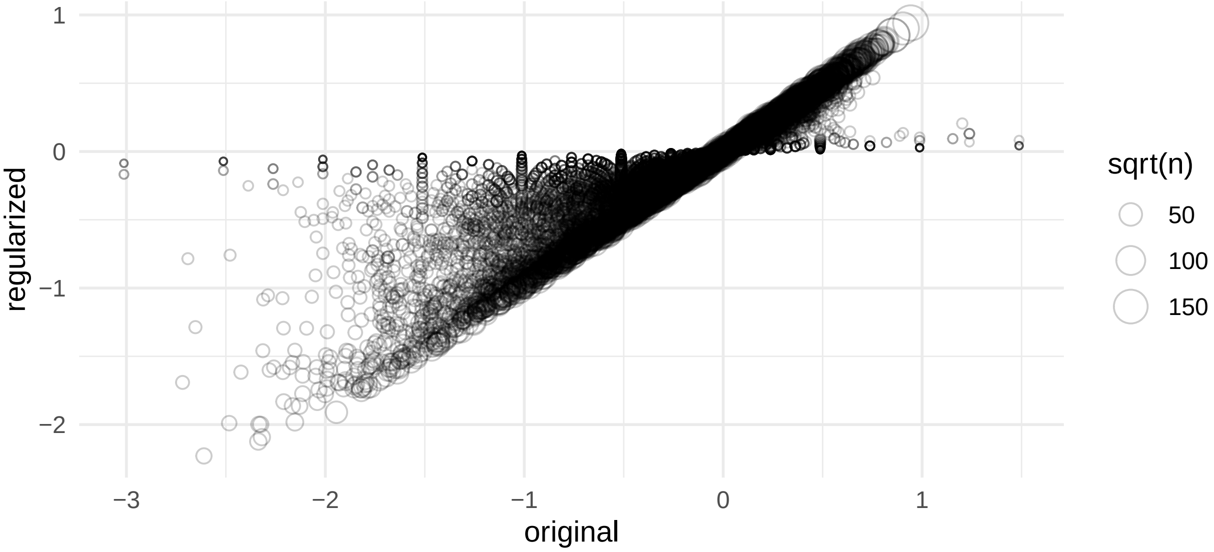


The optimum value for our $Bias_{user}$ figure is 0. Interesting!

Let's check both values against the *test* set and see how it affects our RMSE.

Method	RMSE
Mean Rating (Naive)	1.0612018
Movie Effect Model	0.9439087
User + Movie Effect Model	0.8850398
User + Regularized Movie Effects	0.8839696
Regularized User + Regularized Movie Effects	0.8658939

< 0.5	> 5
78	2,084



Better!

As we can see in the plot above, we're shrinking movies with fewer ratings closer to a bias of zero (that is, towards the overall mean of 3.51).

The movie effect and user effect therefore matter more if the movie/user has more ratings - as either sample size increases we become more confident that the the film truly is better/worse than the overall mean or that the user has a tendency to rate films more or less highly than the overall mean.

3.6.1 Final Improvements

Since we're adding two biase terms together, we have a few predicted ratings below 0.5 and above 5.

These are outside of the valid range empirically present in the dataset. There aren't many, but could we get better by limiting these extreme values to the nearest valid rating?

Method	RMSE
Mean Rating (Naive)	1.0612018
Movie Effect Model	0.9439087
User + Movie Effect Model	0.8850398
User + Regularized Movie Effects	0.8839696
Regularized User + Regularized Movie Effects	0.8658939
Reg. User + Reg. Movie Effects (Capped)	0.8657314

Method	RMSE
Final RMSE	0.8657314

4 Conclusion

10 Million ratings were analyzed from the MovieLens 10M dataset, and bias factors were applied to account for the effects unique to individual users and movies. We regularized those effects to shrink smaller sample sizes towards the global average, and finally limited the resulting values to a range beween 0.5 and 5.

The final RMSE achieved in the model was:

This is better than the target of 0.87750.

	1	2	3	4	6	7
Apollo 13 (1995)	NA	NA	NA	5	NA	NA
Batman (1989)	NA	NA	NA	5	NA	NA
Braveheart (1995)	NA	5	4.5	5	NA	NA
Forrest Gump (1994)	5	NA	NA	NA	NA	NA
Fugitive, The (1993)	NA	NA	NA	NA	5	NA
Jurassic Park (1993)	NA	NA	NA	5	NA	NA
Silence of the Lambs, The (1991)	NA	NA	NA	NA	NA	3
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	NA	5	NA	NA	NA	3
Terminator 2: Judgment Day (1991)	5	NA	NA	5	NA	NA

title	genres_Action	genres_Adventure	genres_Comedy
Boomerang (1992)	0	0	1
Net, The (1995)	1	0	0
Outbreak (1995)	1	0	0
Stargate (1994)	1	1	0
Star Trek: Generations (1994)	1	1	0

4.1 Future Considerations

The model may be improved in the future by adjusting or changing the modeling approach altogether. Below are some of these opportunities that were not included in the final model. Some opportunities are mutually exclusive, as they would rely on entirely different methodologies.

4.1.1 Matrix Factorization

We could consider using matrix factorization to build a model.

We would create a matrix with movies in rows and users in columns (or vice versa), with the intersection being a given users's rating for a given film. Here is an example with the most commonly-rated films, with ratings given by the first 7 users.

Since each user doesn't rate every film, this is what's known as a 'sparse' matrix, with many *"NA"* values.

4.1.2 Rounding

We could perhaps further improve the performance of the model by rounding our predictions to the nearest integer, only when predicting ratings from 2003 and earlier.

4.1.3 Genre Effects

A future model may be improved by considering genre effects, especially user+genre effects.

To achive this, we could use *one-hot encoding* to transform our pipe separated "blob" of genres into individual boolean columns that indicate whether a film falls into that genre.

This would permit us to calculate genre biases (dramas tend to be more highly rated than comedies). A more advanced application could perhaps consider user-genre biases, accounting for the fact that certain users prefer certain genres.

4.1.4 Rating Delay

Movies that are rated shortly after they were released seem to have a different bias than movies that were rated long after they were released. In our correlation plot, we observed that `review_delay` was positively correlated with `rating`.

There are many causal factors (e.g. nostalgia, recency bias, selection bias, true decline in quality over time) that could explain this correlation so additional exploration would be needed to understand this relationship and its potential predictive utility.

4.1.5 Machine learning techniques

Furthermore, to develop the final model, we didn't use any CPU or RAM-intensive machine learning models. For this author, such an undertaking proved infeasible due to technological limitations in handling the size of this dataset.

5 References

1. Rafael A. Irizarry (2021), Introduction to Data Science: https://rafalab.github.io/dsbook/
2. Model Fitting and Recommendation Systems Overview: https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+2T2021/block-v1:HarvardX+PH125.8x+2T2021+type@sequential+block@568d02a4412440489621921b87e7536f/block-v1:HarvardX+PH125.8x+2T2021+type@vertical+block@d7b4b0783dd443d1bd14504fe2333c24