MovieLens Project - Machine Learning Submission

HarvardX Data Science Capstone - PH125.9x

Simon Gibson

2023-12-03

Contents

Introduction

For the 9th Course in the HarvardX Data Science course we have been asked to build a movie recommendation system using the MovieLens dataset. This report will cover the initial creation of the data set, exploration of the data, creation and refinement of the algorithm.

This movie recommendation system is similar to systems used by many companies such as Amazon and Netflix to recommend movies, books, and music to customers.

The Movielens data package can be found at the MovieLens homepage.

MovieLens is a project run by GroupLens - a research lab run at the University of Minnesota in North America. MovieLens is a non-commercial collection of movie data and the main set of data contains over 20 million ratings for over 27,000 movies. In this project we are using the 10M dataset.

In order to test the results of the recommendation system we are using the root-mean-square error (RMSE) to measure the difference between the values predicted by the model and the observed values. For this project a RMSE score of less than 0.86490 is the goal.

Methods

The first step is to clear any set variables so we do not introduce anything unexpected into the data we are working with.

Then we install the packages required to manipulate the data.

```
####################################################
# This code is divided into the following sections #
# 1. Install required packages                     #
# 2. edx code for creating data sets               #
# 3. Data set exploration                          #
####################################################

##########################################################
# 1. Install required packages and download data            #
##########################################################


# Note: this process takes a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "https://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "https://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "https://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra", repos = "https://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "https://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "https://cran.us.r-project.org")
if(!require(stringr)) install.packages("scales", repos = "https://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(dplyr)
library(kableExtra)
library(lubridate)
library(scales)
library(stringr)
```

Following that, the data is downloaded and then divided into 2 sets. The first set is used to train the algorithm and the second set is used to validate the algorithm. By dividing the data the problem of over-training and thus producing skewed results can be avoided.

The creation of the 2 sets involves the following steps. Initially required packages are installed if not installed and then loaded. Next the data is downloaded if the zip files are not found. Column names are set and the data is converted into forms more easily processed. Then the data is joined. Finally the joined data is split into 2 sets - the edx set used to train the algorithm and the final_holdout_test set that will be used to validate the algorithm and calculate the final RMSE score.

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
```

```r
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later

# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.9, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

##########################################
```

```
# End of edx code                              #
################################################
```

Data Exploration

To start with we use the head command to view the first 10 rows of data.

Table 1: EDX Dataset Overview - First 10 Rows

	userId	movieId	rating	timestamp	title	genres
2	1	185	5.0	838983525	Net, The (1995)	Action|Crime|Thriller
20	1	589	5.0	838983778	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
23	2	110	5.0	868245777	Braveheart (1995)	Action|Drama|War
34	2	786	3.0	868244562	Eraser (1996)	Action|Drama|Thriller
49	3	1252	4.0	1133571071	Chinatown (1974)	Crime|Film-Noir|Mystery|Thriller
55	3	1597	4.5	1133571226	Conspiracy Theory (1997)	Drama|Mystery|Romance|Thriller
78	4	39	3.0	844417037	Clueless (1995)	Comedy|Romance
83	4	165	5.0	844416699	Die Hard: With a Vengeance (1995)	Action|Crime|Thriller
87	4	266	5.0	844417070	Legends of the Fall (1994)	Drama|Romance|War|Western
91	4	329	5.0	844416796	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

Looking at the first 5 rows of the data in the edX data set we can see the columns we have to work with - userId, movieId, rating, time stamp, title and genre.

Some initial areas of interest here are the time stamp and genres columns. As time passes do movies get higher ratings?

If we take the example of literature, works such as those by the likes of Homer and Shakespeare survive while over time lesser works are weeded out. Possibly there is some survivability bias that means that movies that continue being reviewed are ones that people have enjoyed and have been recommended, for example through word of mouth or via similar recommendation engines.

The genre column also shows collections of genre keywords, rather than single genres. These collections could also prove to be useful.

Data Summary

Next we can use the summary command to produce result summaries of the results of various model fitting functions.

Table 2: EDX Dataset Summary

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :8.229e+08	Length:1041390	Length:1041390
1st Qu.:18059	1st Qu.: 612	1st Qu.:3.000	1st Qu.:9.456e+08	Class :character	Class :character
Median :35682	Median : 1777	Median :4.000	Median :1.033e+09	Mode :character	Mode :character
Mean :35849	Mean : 4145	Mean :3.515	Mean :1.031e+09	NA	NA
3rd Qu.:53630	3rd Qu.: 3617	3rd Qu.:4.000	3rd Qu.:1.126e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

As we can see from the summary, from a statistical perspective in the current form, the most useful column is the rating row. The time stamp row is in Unix epoch time (seconds from the 1st of January 1970) so that will need to be converted to a human readable format if that is found to be useful.

Data Column Counts

The following table shows the distinct number of User IDs, Movie IDs, Titles, and Genres. The last column is a check for any unset variables. This will return TRUE if present and FALSE if not.

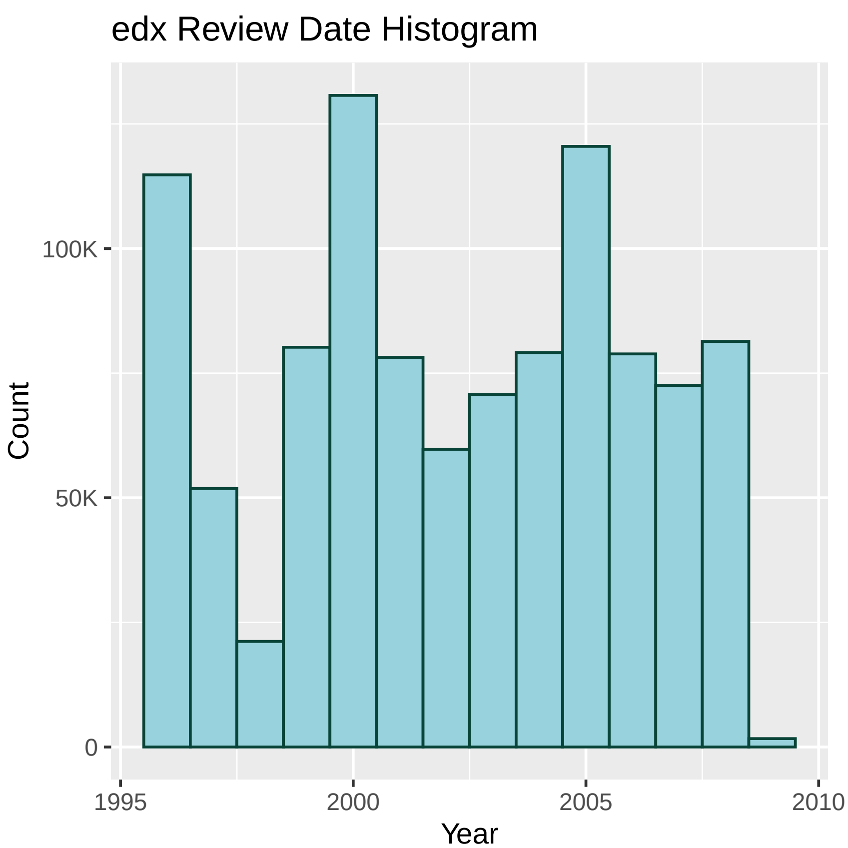
Table 3: Summary of Movielens Data Set

Users	MoviesIds	Titles	Genres	MissingValues
69878	10677	10676	797	FALSE

The number of movies reviewed is higher than the number of reviewers. Also we can see that the number of genres is quite large due to the usage of different arrays of keywords to describe the movies. Also we can see that there are no “Not Available” or missing values.

Ratings Date Range

If we convert the time stamps, we can see that the oldest review is dated 1996-01-29 13:00:00 and the most recent time stamp is 2009-01-05 17:51:47. Given this range it is likely the majority of participants providing ratings were either Baby Boomers or Generation X. Ratings may possibly exhibit generational bias although investigation of this is beyond the scope of the current piece of work.

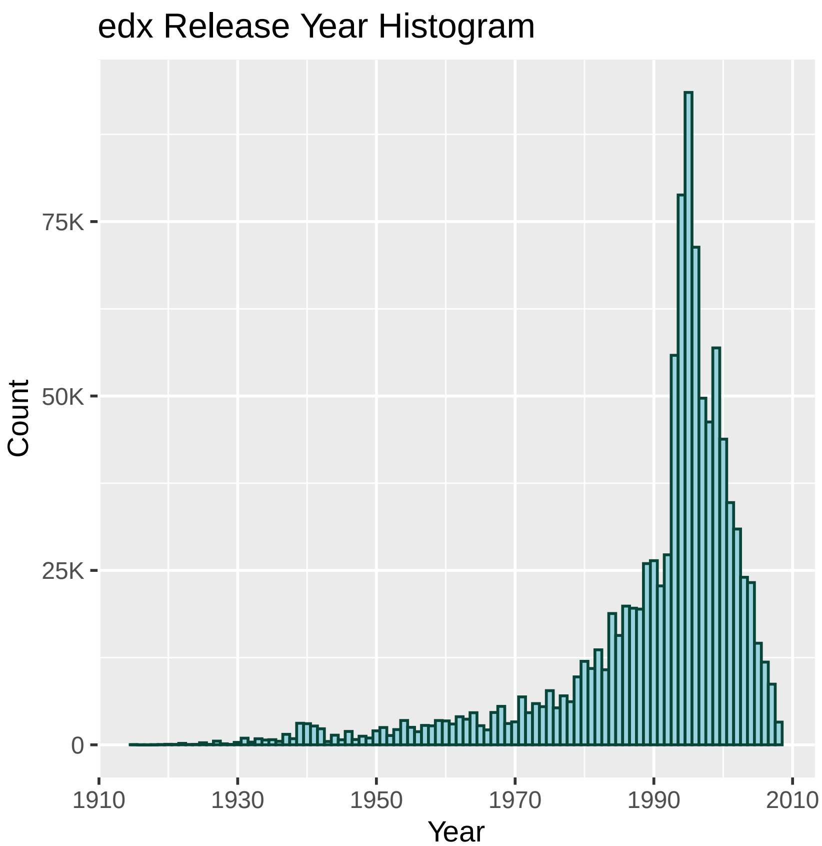


Movie Release Date Range

If we extract the release year from the title column we find that the earliest movie reviewed was released in 1915 and the most recently reviewed movie was released in 2008. This makes sense given the final review in the data set was received at 2009-01-05 17:51:47.

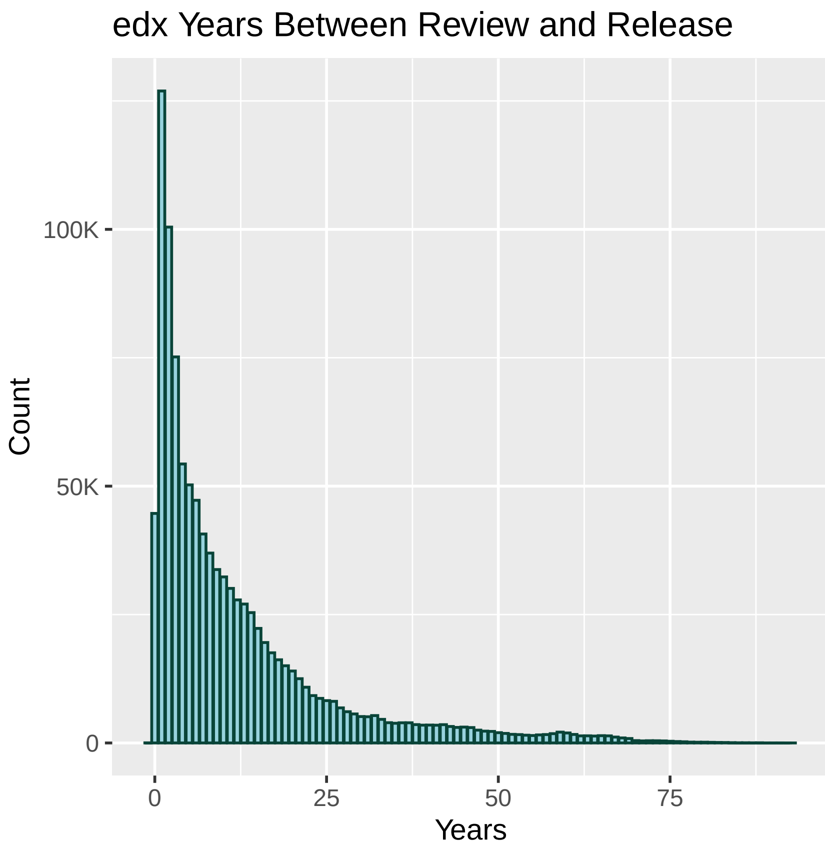
During the time the dataset was collated there were 2 main ways that people were exposed to movies - at theaters and at home on video cassette. Theaters predominantly showed new releases with occasional film festivals and late night showings of classic films such as Clockwork Orange and The Rocky Horror Picture Show.

Video Cassettes were rented from local video stores and were a mix of recent releases and classic films. Video rental stores were limited by floor space to the number of movies that they could offer. This meant that video rental stores focused on offering more mainstream titles within each genre. Chris Anderson in his book The Long Tail: Why the Future of Business Is Selling Less of More wrote about how with the internet and the use of warehousing, businesses were able to offer larger selections than is possible with traditional brick and mortar stores. It is likely that due to this effect a large number of movies have been lost to movie recommendation systems such as the one used in this project.



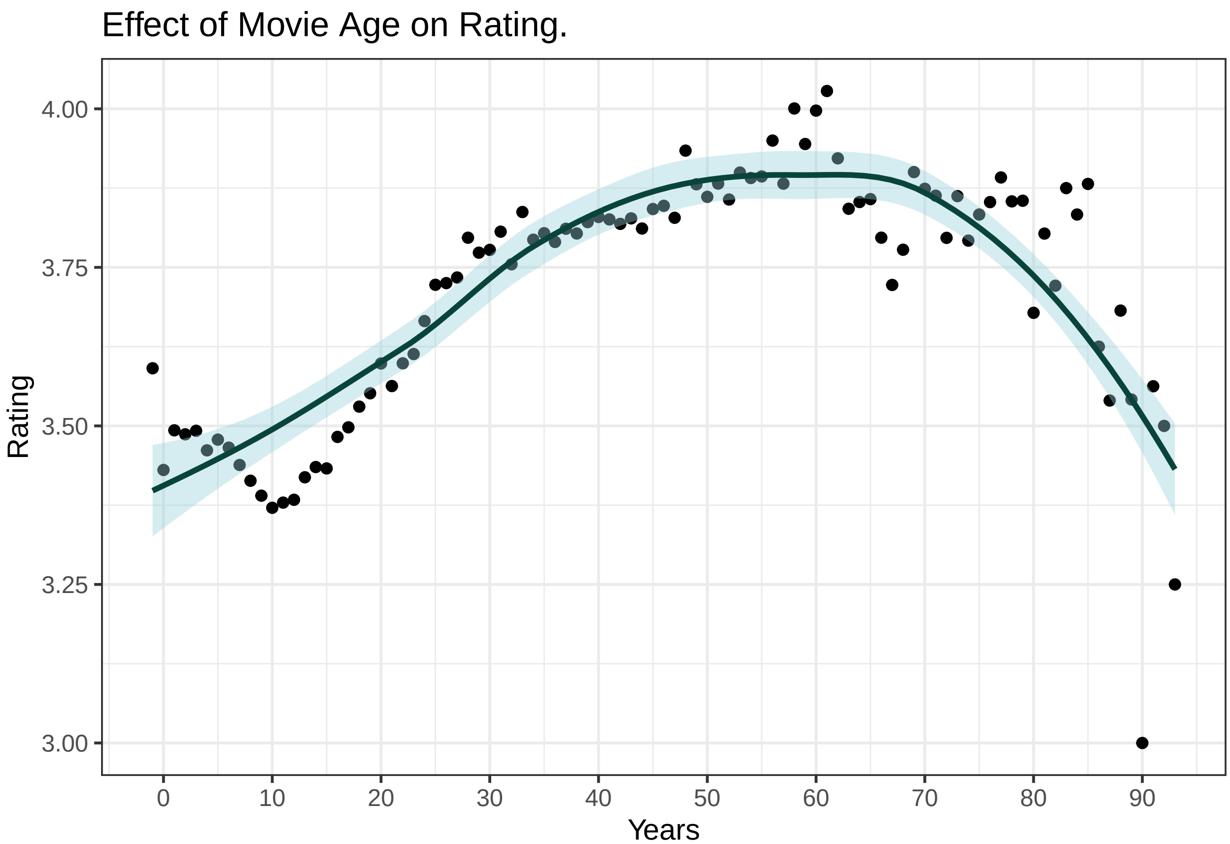
Movie Rating and Release Relationshiop

If we look at the relationship between rating and the length of time between the release of the movie and the review we get the following graph. Interestingly it is almost the opposite of the Release Year histogram.



From the summary output above we saw that ratings have a mean of 3.515 and a median value of 4.

If we combine the age of the movie with the mean rating we get the following graph which shows that older movies have higher average ratings. These movies would predominately be the older movies offered by video stores or exhibited during film festivals. Due to long tail effects these higher ratings are to be expected.

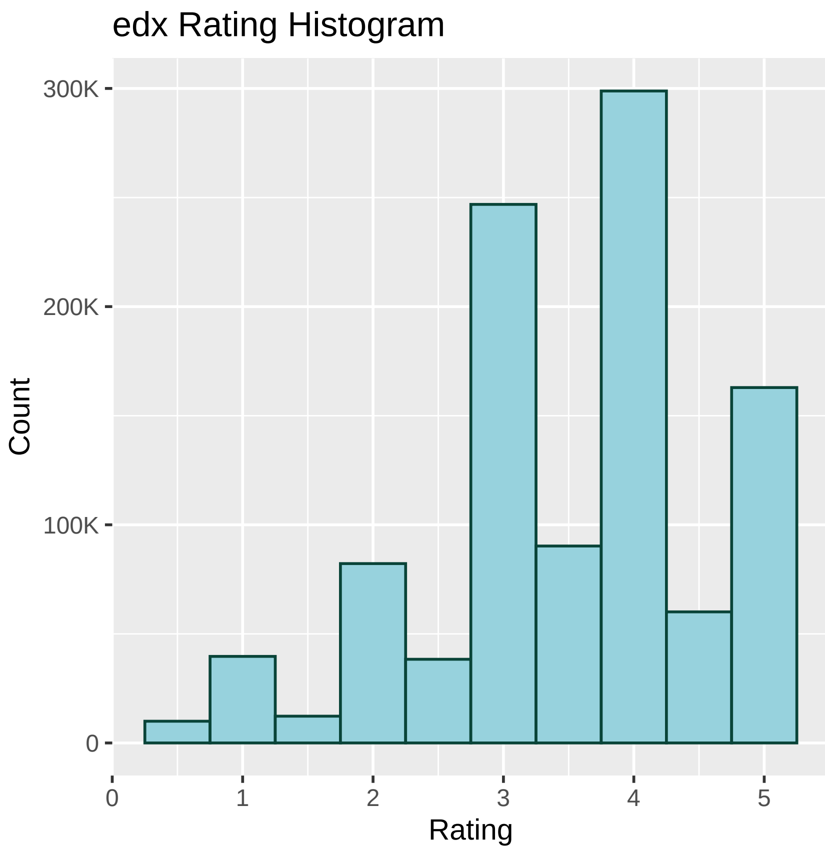


The following table shows the count of ratings. Whole numbers are much more commonly chosen when rating movies than decimal ratings.

Table 4: Rating Distribution

Var1	Freq
0.5	9961
1	39676
1.5	12271
2	82214
2.5	38355
3	246853
3.5	90259
4	298828
4.5	60085
5	162888

From this we can see that people are more likely to rate movies in whole numbers. If we plot this as a graph it is much more evident.



Whole number ratings

Now we will look at the data to see if rating with whole numbers compared to decimals has any impact.

First whole numbers - the subset of the edx dataset that has ratings 1, 2, 3, 4, 5:

Table 5: Whole Number Ratings

Users	MoviesIds	Titles	Genres
68928	10145	10144	784

Decimal Point Ratings

Then the decimal ratings - the subset of the edx dataset with ratings 0.5, 1.5, 2.5, 3.5 or 4.5:

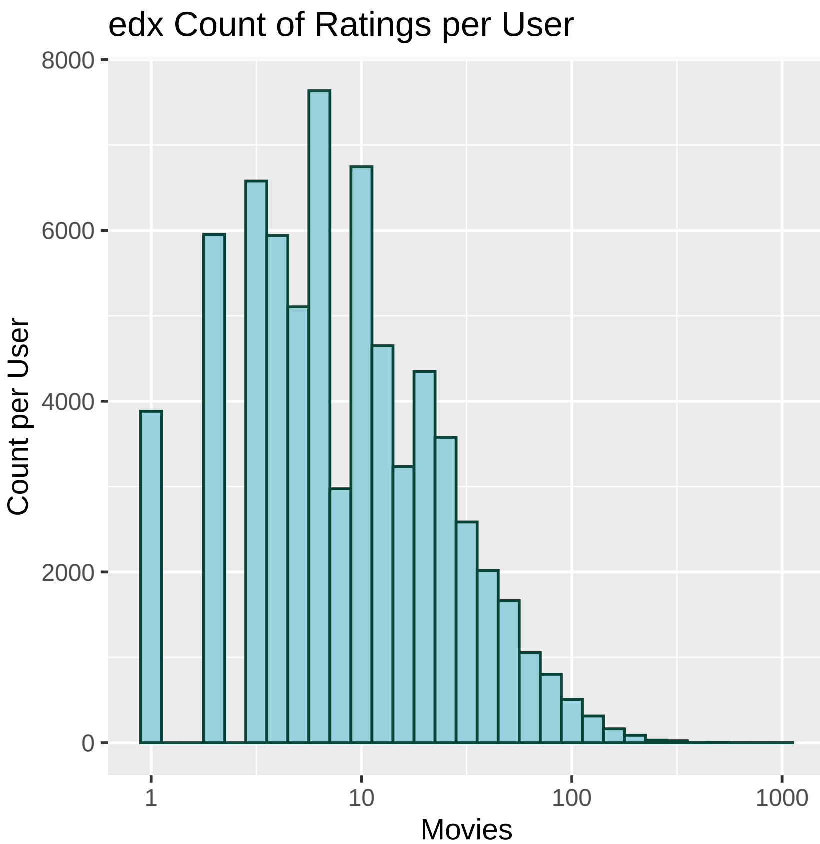
Table 6: Decimal Point Ratings

Users	MoviesIds	Titles	Genres
22674	9119	9118	770

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   2.500   3.500   3.345   4.500   4.500
```

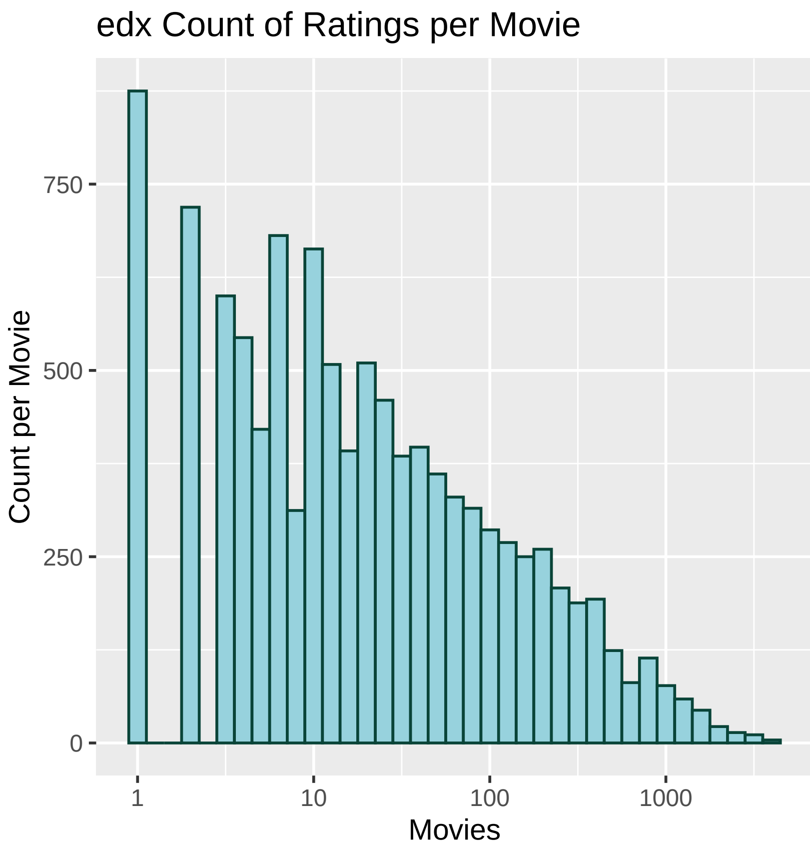
Ratings Per User

Now we turn to the count of ratings per user.

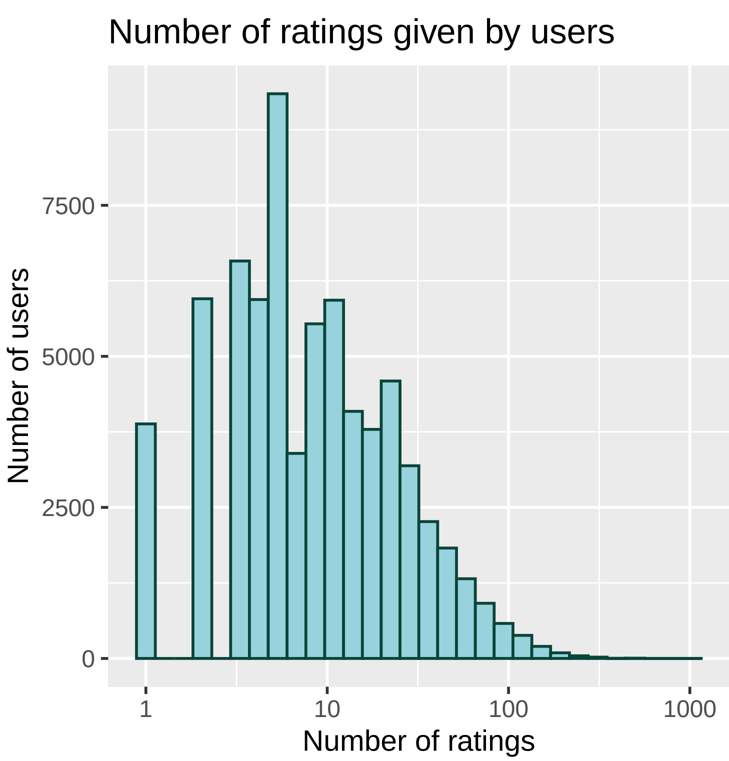


Ratings Per Movie

Again we can see that some movies are more popular than others and therefore have more reviews than less popular films.



If we look at the average number of films reviewed by each reviewer we get the following results.



Model Investigation

RMSE Function and Target

As mentioned in the introduction, we have been asked to use a RMSE function to test our machine learning algorithms.

This is defined as

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

where N is the number of users/movie combinations, $y_{u,i}$ is the rating for for movie i by user u and $\hat{y}_{u,i}$ is our prediction.

The RMSE function can be written as follows.

```
#RMSE calculation function
RMSE <- function(predicted_ratings, true_ratings){
  sqrt(mean((predicted_ratings - true_ratings)^2))

}
```

The goal of this paper is to create a model that will produce a result lower than the Target RMSE:

Table 7: RMSE Results

method	RMSE
Target RMSE	0.8649

Naive RMSE

To begin with we find the mean and then use that to find the naive RMSE. This model assumes all differences are the result of random error and is defined as follows:

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

where μ the expected rating and $\epsilon_{u,i}$ is independent error across all movies.

```
#Derive the mean and apply to RMSE function
mu_hat <- mean(edx$rating, na.rm = TRUE)

naive_rmse <- RMSE(edx$rating, mu_hat)
```

Table 8: RMSE Results

method	RMSE
Target RMSE	0.864900
Naive RMSE	1.061233

The result returned using the Naive RMSE is greater than 1 which is well above the target RMSE of less than 0.86490. Therefore we will proceed to work through the findings of our data exploration to find a result that meets the required target.

Movie Effect

During the data exploration it was noted that some movies were rated higher than others. Here we test the impact of this by calculating the difference between the mean rating of the movie and the overall mean. This can be expressed as follows where b_i represents bias for each movie i:

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

```
#use mean derived above
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(lse = mean(rating - mu_hat))

predicted_ratings <- mu_hat + edx %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(lse)

model_1_rmse <- RMSE(predicted_ratings, edx$rating)
```

Table 9: RMSE Results

method	RMSE
Target RMSE	0.864900
Naive RMSE	1.061233
Group By MovieID	0.940688

Adding in the Movie effect brings the RMSE below 1 which is an improvement however it is still above the Target RMSE.

User ID Effect

Next we will apply the findings of the data exploration in terms of the effect of the User bias on the model. Each individual has their own preferences and this addition will allow for some individuals rating some movies higher than others or lower. We do not have detailed demographic data to apply to this model however note that this would be highly beneficial.

Adding the user effect b_u to our model gives us:

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

```
#compute an approximation by computing mu and b_i and estimating b_u as the average
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - lse))

predicted_ratings <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + lse + b_u) %>%
  pull(pred)
```

Table 10: RMSE Results

method	RMSE
Target RMSE	0.8649000
Naive RMSE	1.0612325
Group By MovieID	0.9406880
Group By MovieID + UserID	0.8285405

The result of "Group By MovieID" combined with "Group by UserID" has returned an RMSE lower than the target score.

Finally let us run this model against the whole number subset of data and against the decimal only set of data to see if this has any impact.

Whole Number Data Subset

```
mu_hat <- mean(edx1$rating, na.rm = TRUE)
movie_avgs <- edx1 %>%
  group_by(movieId) %>%
  summarize(lse = mean(rating - mu_hat))

predicted_ratings <- mu_hat + edx1 %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(lse)

model_1W_rmse <- RMSE(predicted_ratings, edx1$rating)
rmse_results <- rmse_results %>% add_row(method = "Whole Number - Group By MovieID", RMSE = model_1W_rms

user_avgs <- edx1 %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - lse))

predicted_ratings <- edx1 %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + lse + b_u) %>%
  pull(pred)

model_2W_rmse <- RMSE(predicted_ratings, edx1$rating)

rmse_results <- rmse_results %>% add_row(method = "Whole Number - Group By MovieID + UserID", RMSE = mod
kable(rmse_results, caption = "RMSE Results", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Running against the subset of Ratings that use whole number only returns RMSE results statistically on par with the larger set of data.

Decimal Number Data Subset

```
mu_hat <- mean(edx0.5$rating, na.rm = TRUE)
movie_avgs <- edx0.5 %>%
```

Table 11: RMSE Results

method	RMSE
Target RMSE	0.8649000
Naive RMSE	1.0612325
Group By MovieID	0.9406880
Group By MovieID + UserID	0.8285405
Whole Number - Group By MovieID	0.9393667
Whole Number - Group By MovieID + UserID	0.8239745

```
  group_by(movieId) %>%
  summarize(lse = mean(rating - mu_hat))

predicted_ratings <- mu_hat + edx0.5 %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(lse)

model_1.5_rmse <- RMSE(predicted_ratings, edx0.5$rating)
rmse_results <- rmse_results %>% add_row(method = "Decimal Number - Group By MovieID", RMSE = model_1.5

user_avgs <- edx0.5 %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - lse))

predicted_ratings <- edx0.5 %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + lse + b_u) %>%
  pull(pred)

model_2.5_rmse <- RMSE(predicted_ratings, edx0.5$rating)

rmse_results <- rmse_results %>% add_row(method = "Decimal Number - Group By MovieID + UserID", RMSE = 
kable(rmse_results, caption = "RMSE Results", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 12: RMSE Results

method	RMSE
Target RMSE	0.8649000
Naive RMSE	1.0612325
Group By MovieID	0.9406880
Group By MovieID + UserID	0.8285405
Whole Number - Group By MovieID	0.9393667
Whole Number - Group By MovieID + UserID	0.8239745
Decimal Number - Group By MovieID	0.9005922
Decimal Number - Group By MovieID + UserID	0.7557891

Running against the subset of Ratings that use decimal numbers only returns RMSE results statistically more accurate when compared with either the complete data set or the set of whole number only results. This is possibly caused by the people who are prepared to use decimal ratings having greater accuracy in their ratings.

Validation

To validate this result we need to run the final model against the final_holdout_data set.

```r
#remove whole number ratings
seq0.5 <- seq(0.5,4.5,1)
final_holdout_test <- final_holdout_test[final_holdout_test$rating %in% seq0.5,]

#Derive the mean
mu_hat <- mean(final_holdout_test$rating, na.rm = TRUE)

#calculate Movie Effect
movie_avgs <- final_holdout_test %>%
  group_by(movieId) %>%
  summarize(lse = mean(rating - mu_hat))

predicted_ratings <- mu_hat + final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(lse)

# compute an approximation by computing mu and b_i and estimating b_u as the average

user_avgs <- final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - lse))

predicted_ratings <- final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + lse + b_u) %>%
  pull(pred)


final_holdout_test_RMSE <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_results <- rmse_results %>% add_row(method = "Final Holdout Test", RMSE = final_holdout_test_RMSE)
```

Results

Final RMSE is 0.8073644 which is lower than the Target RMSE 0.8649.

Conclusion

Summary

The project has delivered a working algorithm that meets the requirements of the course.

Table 13: RMSE Results

method	RMSE
Target RMSE	0.8649000
Naive RMSE	1.0612325
Group By MovieID	0.9406880
Group By MovieID + UserID	0.8285405
Whole Number - Group By MovieID	0.9393667
Whole Number - Group By MovieID + UserID	0.8239745
Decimal Number - Group By MovieID	0.9005922
Decimal Number - Group By MovieID + UserID	0.7557891
Final Holdout Test	0.8073644

Limitations

With this project hardware limitations were often reached resulting in rstudio crashing frequently. Reducing the data set size helped but iterating the model was limited due to the time required to run each test.

Future work

It would be good to add back into the data set movie ratings rated by UserId where user has used decimal ratings for at least one movie to see if this has any impact.

Regularisation can also be applied.

Switching to GPU processing, possibly leveraging CUDA would be interesting and enjoyable.

Automated testing would be useful. Exploration of movies that didn't make it into video stores during the 80s and 90s would be interesting in the sense that revisiting the long tail of videos may discover new works that were excluded due to mainstream tastes of that period.

References

1. http://rafalab.dfci.harvard.edu/dsbook/large-datasets.html#recommendation-systems
2. https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+3T2022/block-v1:HarvardX+PH125.8x+3T2022+type@sequential+block@7e7727ce543b4ed6ae6338626862eada/block-v1:HarvardX+PH125.8x+3T2022+type@vertical+block@df3d8a86b43f4247a4dd42bcabb1a663
3. https://anderfernandez.com/en/blog/how-to-code-a-recommendation-system-in-r/
4. Chris Anderson, The Long Tail: Why the Future of Business Is Selling Less of More, Hyperion 2006.
5. Harold Bloom, The Western Canon: The Books and School of the Ages. New York: Harcourt Brace, 1994.