What Movie Should I Watch Tonight?

Jane Farris

12/01/2020

Contents

Overview

This MovieLens project is part of the HarvardX Professional Certificate in Data Science capstone course. It consists of three files: a report in the form of an Rmd file, a report in the form of a PDF document (knit from the Rmd file), and an R script that generates the predicted movie ratings. This report contains the objective, data cleaning, exploratory analysis, data visualization, modeling, results and concluding remarks of the project.

Introduction

One of the most common applications of machine learning is the use of recommendation systems to predict a rating or preference that a user will give to an item. Recommendation systems make specific suggestions for what a user might enjoy based on previously collected data, ratings and user activity.

In 2006 Netflix released a challenge to the data science community: anyone who can improve our recommendation algorithm by 10% will win a $1M prize. This project is inspired by the 2006 Netflix challenge and details the creation of a movie recommendation system using a version of the MovieLens dataset contained in the dslabs package.

Objective

The goal of this project is to train and develop a machine learning algorithm that strongly predicts a users rating of a movie (ranging from 0.5 to 5 stars), using the inputs of a subset of the MovieLens data. The machine learning techniques will be evaluated using Root Mean Square Error (RMSE) in order to see how far off the model predictions are from the observed user ratings, i.e. how accurate they are.

This report contains the seven models developed to predict a users movie rating using various machine learning techniques and input variables. The accuracy of each model is assessed and reported in the form of RMSE and then compared in order to find the best movie recommendation algorithm.

The Dataset

MovieLens 10M datasets: https://grouplens.org/datasets/movielens/10m/ http://files.grouplens.org/datasets/movielens/ml-10m.zip (zip file)

First we must create our train and test sets from the MovieLens data set using the tidyverse and caret packages. Our final validation (test) set will be 10% of our original MovieLens data set.

The resulting datasets are as follows and are of the following dimensions.

```
#Structure and size of the edx dataset used to train ML algorithms
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  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 (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
dim(edx)
```

```
## [1] 9000055       6
```

```
#Structure and size of the validation dataset used as a final hold-out test set
str(validation)
```

```
## Classes 'data.table' and 'data.frame':   999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Roman
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
dim(validation)
```

```
## [1] 999999      6
```

As we can see from the above code, the rows in both the edx and validation datasets represent a users movie rating for a single movie. The dataset also includes six variables:

"rating" : the outcome we are predicting, a users movie rating between 0 and 5 (continuous)

"userID" : unique ID number given to each user (discrete)

"movieID" : unique ID number given to each movie (discrete)

"timestamp" : the date and time the rating was given (discrete)

"title" : movie title and year the movie was released (nominal)

"genres" : names of the genre(s) of the given movie (nominal)

Methods/Analysis

This section of the report contains the data cleaning, data exploration, data visualization and modelling techniques used.

Data Cleaning

Now we must make sure that our edx data is in a clean, digestible format with no missing values or errors.

First we must check for any missing values in our edx and validation datasets.

```
sum(is.na(edx))
```

```
## [1] 0
```

```
sum(is.na(validation))
```

```
## [1] 0
```

```
summary(edx) #alternate way to see there are no missing values in the edx dataset
summary(validation) #alternate way to see there are no missing values in the validation dataset
```

In order to make our data more readable, we must convert the "timestamp" column into a universal format. We can use the as_datetime() function within the lubridate package in order to convert the timestamp from seconds since January 1, 1970 to an appropriate form. The result is in a new column named "date".

```
library(lubridate)
edx <- mutate(edx, date = date(as_datetime(timestamp, origin="1970-01-01")))
#The date of rating is now in the correct form YYYY/MM/DD
head(edx$date)
```

```
## [1] "1996-08-02" "1996-08-02" "1996-08-02" "1996-08-02" "1996-08-02"
## [6] "1996-08-02"
```

```
#Drop the timestamp column
edx <- edx %>% select(-timestamp)

validation <- validation %>% mutate(validation,date = date(as_datetime(timestamp, origin="1970-01-01")))
#The date of rating is now in the correct form YYYY/MM/DD
head(validation$date)
```

```
## [1] "1996-08-02" "1996-08-02" "1996-08-02" "1997-07-07" "1997-07-07"
## [6] "1997-07-07"
```

```
#Drop the timestamp column
validation <- validation %>% select(-timestamp)
```

Next, the "title" variable contains both the movie title and the year it was released. We must extract the movie release year and separate it from the movie title by creating a separate column "year_released"

```
#Separate the movie release year and the movie title from the title column
#The last 6 characters of the title string represent the year in brackets
#Thus the indices -5 to -2 represent the release year without brackets
edx <- mutate(edx,year_released = as.numeric(str_sub(title,-5,-2)))
validation <- mutate(validation,year_released = as.numeric(str_sub(title,-5,-2)))

#Remove the year from the title column
library(stringr)
```

```
edx <- mutate(edx,title = str_sub(title,1,-7))
validation <-mutate(validation,title = str_sub(title,1,-7))
```

Now we must check that the transformed timestamps in the the date column actually make sense. In other words, we need to make sure that the converted timestamp date doesn't fall before the movie release date.

```
#We can see that there are no ratings that occur before the movie premier date
edx %>% filter(date<year_released)
```

```
## Empty data.table (0 rows and 7 cols): userId,movieId,rating,title,genres,date...
```

```
validation %>% filter(date<year_released)
```

```
## Empty data.table (0 rows and 7 cols): userId,movieId,rating,title,genres,date...
```

Looking at the "genres" column, we see that there are sometimes multiple genres per movie. We can also see that there are 797 unique genre combinations each movie could have. In order to see what genres are most rated and potentially impact the rating prediction of a movie, we must separate each genre type. Instead of having one column with multiple genres listed, we will have dummy variable columns for each of the 20 distinct genres.

```
#All distinct combinations of genre types in the dataset
all_genre_combinations <- unique(edx$genres)

#Detect genre types based on the separater "|"
genre_types <- unlist(strsplit(edx$genres, split = "\\|"))
#Keep only one of each genre
genre_list <- unique(genre_types)
#We can see there are 20 different genre types
genre_list
```

```
##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

```
#Create 20 binary columns corresponding to each genre type in the genre_list object
edx <- edx %>% mutate(comedy=ifelse(str_detect(genres, "Comedy"),1,0),
                      romance=ifelse(str_detect(genres, "Romance"),1,0),
                      action=ifelse(str_detect(genres, "Action"),1,0),
                      crime=ifelse(str_detect(genres, "Crime"),1,0),
                      thriller=ifelse(str_detect(genres, "Thriller"),1,0),
                      drama=ifelse(str_detect(genres, "Drama"),1,0),
                      scifi=ifelse(str_detect(genres, "Sci-Fi"),1,0),
                      adventure=ifelse(str_detect(genres, "Adventure"),1,0),
                      children=ifelse(str_detect(genres, "Children"),1,0),
                      fantasy=ifelse(str_detect(genres, "Fantasy"),1,0),
                      war=ifelse(str_detect(genres, "War"),1,0),
                      animation=ifelse(str_detect(genres, "Animation"),1,0),
                      musical=ifelse(str_detect(genres, "Musical"),1,0),
                      western=ifelse(str_detect(genres, "Western"),1,0),
                      mystery=ifelse(str_detect(genres, "Mystery"),1,0),
                      filmnoir=ifelse(str_detect(genres, "Film-Noir"),1,0),
                      horror=ifelse(str_detect(genres, "Horror"),1,0),
```

```
                      documentary=ifelse(str_detect(genres, "Documentary"),1,0),
                      imax=ifelse(str_detect(genres, "IMAX"),1,0),
                      none=ifelse(str_detect(genres, "(no genres listed)"),1,0))

validation <- validation %>% mutate(comedy=ifelse(str_detect(genres, "Comedy"),1,0),
                      romance=ifelse(str_detect(genres, "Romance"),1,0),
                      action=ifelse(str_detect(genres, "Action"),1,0),
                      crime=ifelse(str_detect(genres, "Crime"),1,0),
                      thriller=ifelse(str_detect(genres, "Thriller"),1,0),
                      drama=ifelse(str_detect(genres, "Drama"),1,0),
                      scifi=ifelse(str_detect(genres, "Sci-Fi"),1,0),
                      adventure=ifelse(str_detect(genres, "Adventure"),1,0),
                      children=ifelse(str_detect(genres, "Children"),1,0),
                      fantasy=ifelse(str_detect(genres, "Fantasy"),1,0),
                      war=ifelse(str_detect(genres, "War"),1,0),
                      animation=ifelse(str_detect(genres, "Animation"),1,0),
                      musical=ifelse(str_detect(genres, "Musical"),1,0),
                      western=ifelse(str_detect(genres, "Western"),1,0),
                      mystery=ifelse(str_detect(genres, "Mystery"),1,0),
                      filmnoir=ifelse(str_detect(genres, "Film-Noir"),1,0),
                      horror=ifelse(str_detect(genres, "Horror"),1,0),
                      documentary=ifelse(str_detect(genres, "Documentary"),1,0),
                      imax=ifelse(str_detect(genres, "IMAX"),1,0),
                      none=ifelse(str_detect(genres, "(no genres listed)"),1,0))
```

Note that movie title beginning with "The" are formatted incorrectly in the title column. For instance, the movie title "The Flintstones" is inputted as "Flintstones, The". Finally, to make the title column even more readable, we must detect these cases and fix them.

```
#Remove the extra space at the end of all movie titles
edx <- mutate(edx, title=str_trim(title, side = c("right")))
validation <- mutate(validation, title=str_trim(title,side=c("right")))
#Relocate "The" if it's in the movie title
edx <- mutate(edx, title = sub('^(.*), The', 'The \\1', title))
validation <- mutate(validation, title = sub('^(.*), The', 'The \\1', title))
```

Final Cleaned Dataset

```
#Final clean dataset
head(edx)
```

```
##    userId movieId rating                title                        genres
## 1:      1     122      5              Boomerang                Comedy|Romance
## 2:      1     185      5                The Net          Action|Crime|Thriller
## 3:      1     292      5               Outbreak  Action|Drama|Sci-Fi|Thriller
## 4:      1     316      5               Stargate       Action|Adventure|Sci-Fi
## 5:      1     329      5 Star Trek: Generations Action|Adventure|Drama|Sci-Fi
## 6:      1     355      5        The Flintstones       Children|Comedy|Fantasy
##          date year_released comedy romance action crime thriller drama scifi
## 1: 1996-08-02          1992      1       1      0     0        0     0     0
## 2: 1996-08-02          1995      0       0      1     1        1     0     0
## 3: 1996-08-02          1995      0       0      1     0        1     1     1
## 4: 1996-08-02          1994      0       0      1     0        0     0     1
```

```
## 5: 1996-08-02          1994      0       0      1     0        0     1     1
## 6: 1996-08-02          1994      1       0      0     0        0     0     0
##    adventure children fantasy war animation musical western mystery filmnoir
## 1:         0        0       0   0         0       0       0       0        0
## 2:         0        0       0   0         0       0       0       0        0
## 3:         0        0       0   0         0       0       0       0        0
## 4:         1        0       0   0         0       0       0       0        0
## 5:         1        0       0   0         0       0       0       0        0
## 6:         0        1       1   0         0       0       0       0        0
##    horror documentary imax none
## 1:      0           0    0    0
## 2:      0           0    0    0
## 3:      0           0    0    0
## 4:      0           0    0    0
## 5:      0           0    0    0
## 6:      0           0    0    0
```

Now our data is in the clean, tidy format with no missing values and meaningful columns, thus it is ready for exploration, visualization and further analysis. Our final dataset includes 27 variables:

"rating" : the outcome we are predicting, a users movie rating between 0 and 5 (continuous)

"userID" : unique ID number given to each user (discrete)

"movieID" : unique ID number given to each movie (discrete)

"title" : movie title (nominal)

"genres" : names of the genre(s) of the given movie (nominal)

"date" : date the rating was given in the form YYYY/MM/DD (discrete)

"year_released" : year the movie was released (discrete)

"comedy",**"romance"**,**"action"**,**"crime"**,**"thriller"**,**"drama"**,**"scifi"**,**"adventure"**,**"children"**,**"fantasy"**,**"war"**,**"an**
: these 20 binary columns indicate whether the movie is listed in the corresponding genre or not

Data Exploration & Visualization

We can see that our dataset contains 10,677 different movies that are being rated. Also, there are 69,878 different users rating these movies.

```
#Number of distinct movies in the dataset
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
#Number of distinct users in the dataset
n_distinct(edx$userId)
```

```
## [1] 69878
```

We can also see that our dataset includes ratings that were given in between January 9, 1995 and January 5, 2009, thus our rating period was recorded over the span of approximately 14 years.

```
#first rating date
max(edx$date)
```

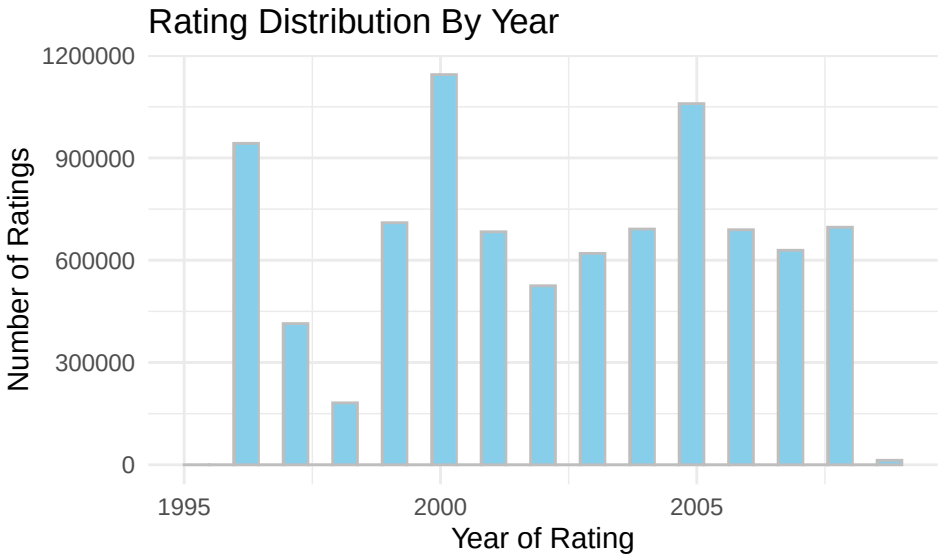
```
## [1] "2009-01-05"
```

```
#last rating date
min(edx$date)
```

```
## [1] "1995-01-09"
```

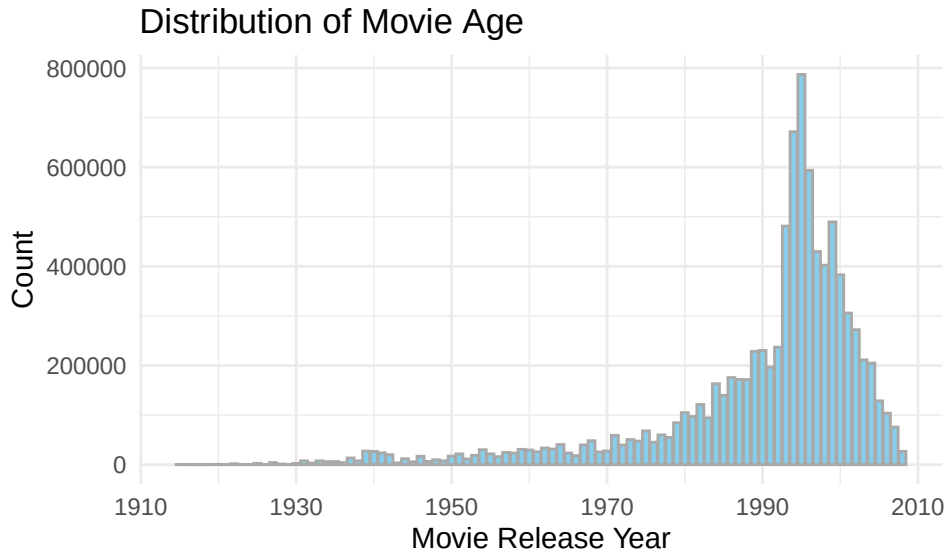
Data Exploration: Year of Rating

First, we can examine the distribution of the year a movie was rated, using the timestamp data we were given. This graph shows that 1996, 2000 and 2005 were the most popular years for users to give movie ratings. Also note 1998 had significantly less movie ratings than the rest of the years in the 14 year range (2009 has the least amount of ratings, however only 5 days of the year were recorded in the dataset, explaining the low rating volume).



Data Exploration: Movie Release Date (Movie Age)

Another date we can explore is the year a movie was released, i.e. the age of the movie. We can see the distribution of movie ratings by movie age. We can see that this distribution is left skewed, with movies released between 1993 and 2000 having the most amount of ratings in our dataset. On the other hand, movies released between 1915-1970 and 2003-2008, have a much lower amount of ratings. This indicates a trend that movies released in the last 5 years tend to have fewer ratings since they have just been released, however they still have significantly more ratings than older movies released prior to 1970. Thus movies released between 1970 and 2003 have a much greater amount of ratings in our dataset.

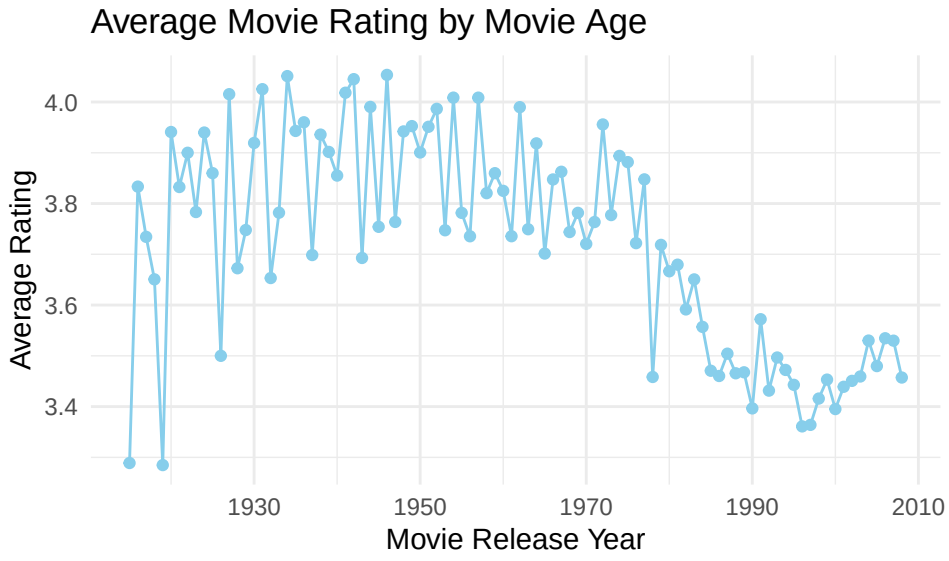


There are 94 different movie ages in our data set and we can calculate the average movie rating for each of those movie ages. We see that movies released in 1946 have the highest average rating of approximately 4.05, while movies released in 1919 have the lowest average rating of approximately 3.28. We can see from the graph of movie age versus average rating that more modern movies (movie released after 1980) have much lower ratings.

year_released	rating
1915	3.288889
1916	3.833333
1917	3.734375
1918	3.650685
1919	3.284810
1920	3.940870

year_released	rating
1946	4.053341

year_released	rating
1919	3.28481



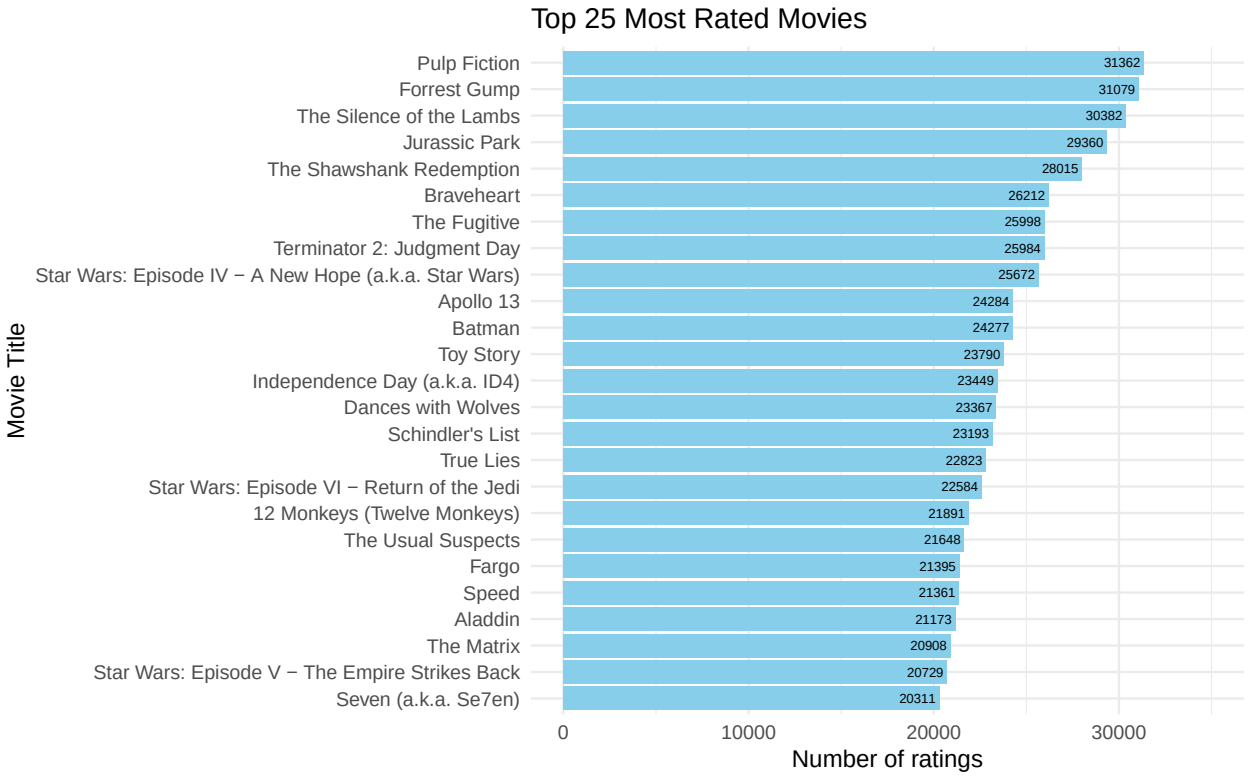
Data Exploration: Movie Titles

Now we can see which movie titles are rated the most, and similarily which ones are rated the least. We can determine that most rated movie is Pulp Fiction, followed closely by Forrest Gump and The Silence of the Lambs. On the other hand we see that there are lots of movies that have been rated only one time, such as The Quarry and Hellhounds on My Trail.

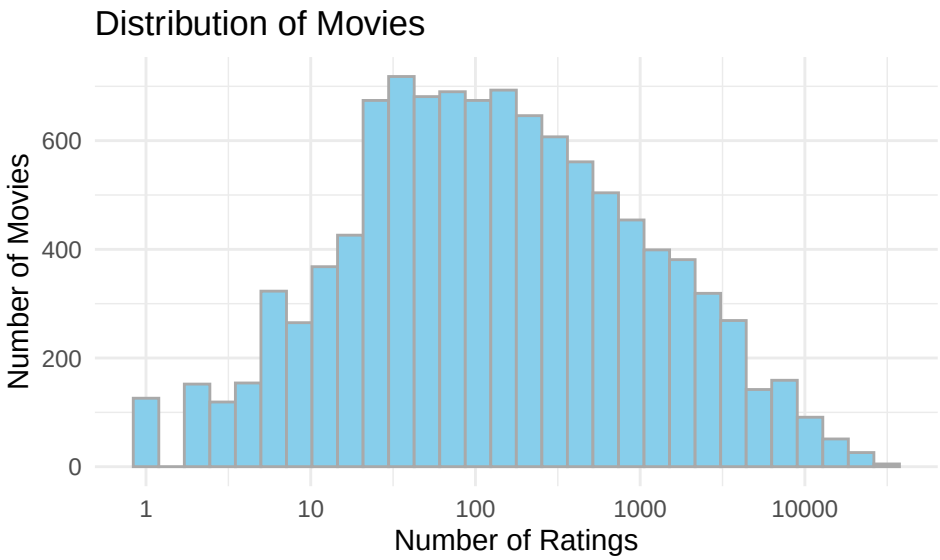
movieId	title	count
296	Pulp Fiction	31362
356	Forrest Gump	31079
593	The Silence of the Lambs	30382
480	Jurassic Park	29360
318	The Shawshank Redemption	28015
110	Braveheart	26212

movieId	title	count
3191	The Quarry	1
3226	Hellhounds on My Trail	1
3234	Train Ride to Hollywood	1
3356	Condo Painting	1
3383	Big Fella	1
3561	Stacy's Knights	1

Here is a visual breakdown of the 25 most rated movies in our dataset.



Further, we know that not every movie is rated the same amount of times, so we can explore the distribution of movies in the following graph. Some more popular movies will have lots of ratings, but other smaller ones will only be watched by few and thus rated less frequently. We can see from the graph that the distribution of movies is roughly symmetric, with most movies being rated 10-1000 times, while movies with over 10,000 ratings are less common.



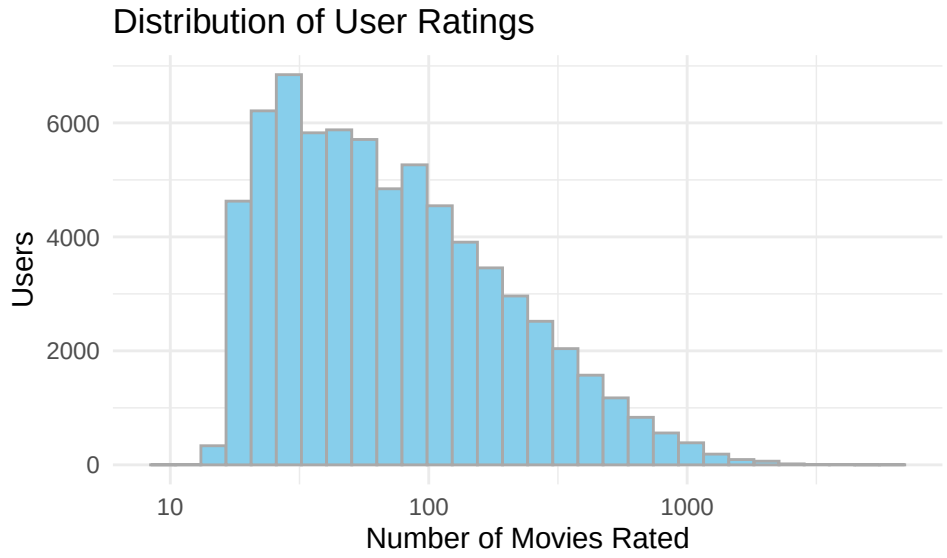
Data Exploration: Users

We can also look at how frequently users rate movies, i.e. the distribution of users. We can see that the distribution is right skewed with a very small percentage of people rating less than 20 movies. Most users rate

30-100 movies, with a much smaller amount of users rating over 1000 movies. The most common amount of movies for a user to rate is around 50 movies. Overall we can see that each user has a different level of activity, some users are more active than others, so we must take that into account when making our predictions.

```
## [1] 71567
```

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

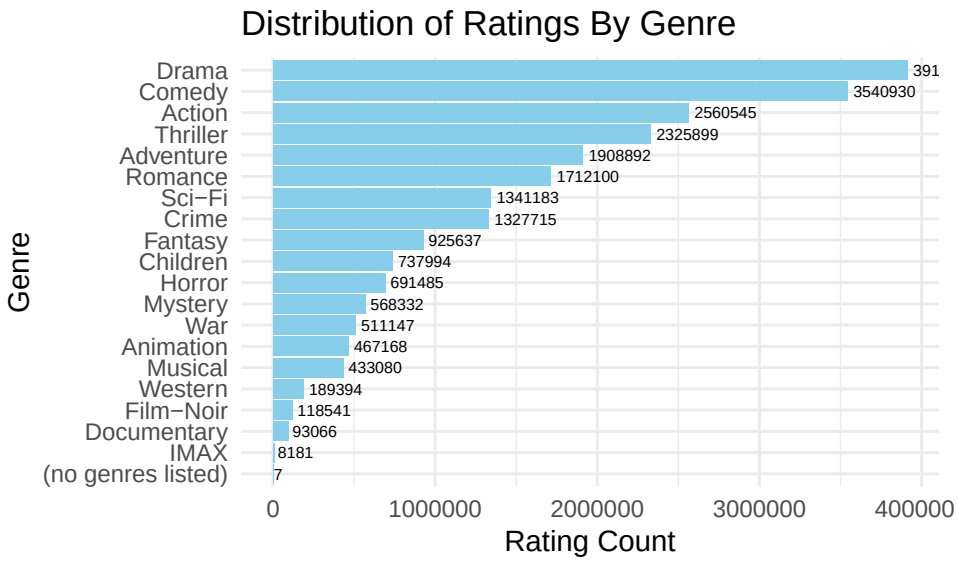


Data Exploration: Genres

We can also see the breakdown of movie ratings by genre. We see that Drama is the largest genre and No Genre is the smallest.

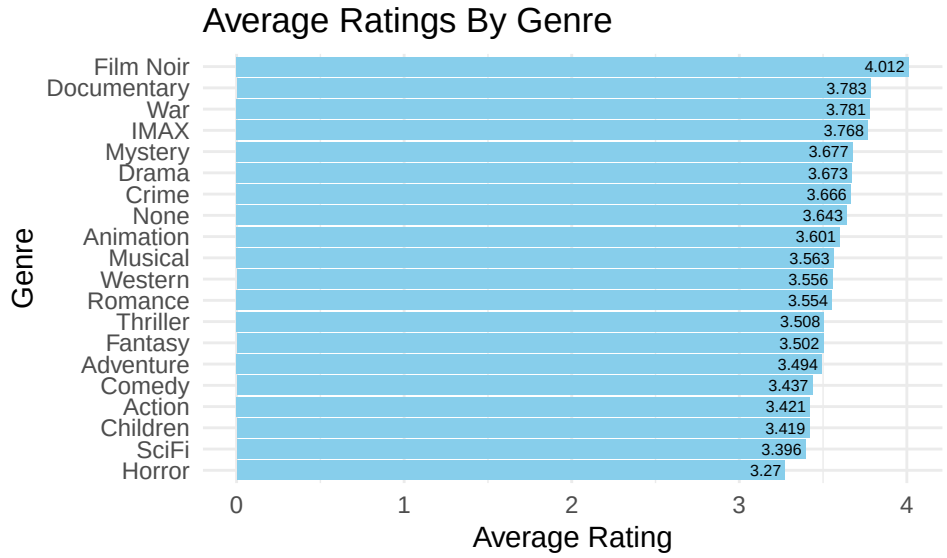
comedy	romance	action	crime	thriller	drama	scifi	adventure	children	fantasy
3540930	1712100	2560545	1327715	2325899	3910127	1341183	1908892	737994	925637

fantasy	war	animation	musical	western	mystery	filmnoir	horror	documentary	imax	none
925637	511147	467168	433080	189394	568332	118541	691485	93066	8181	7

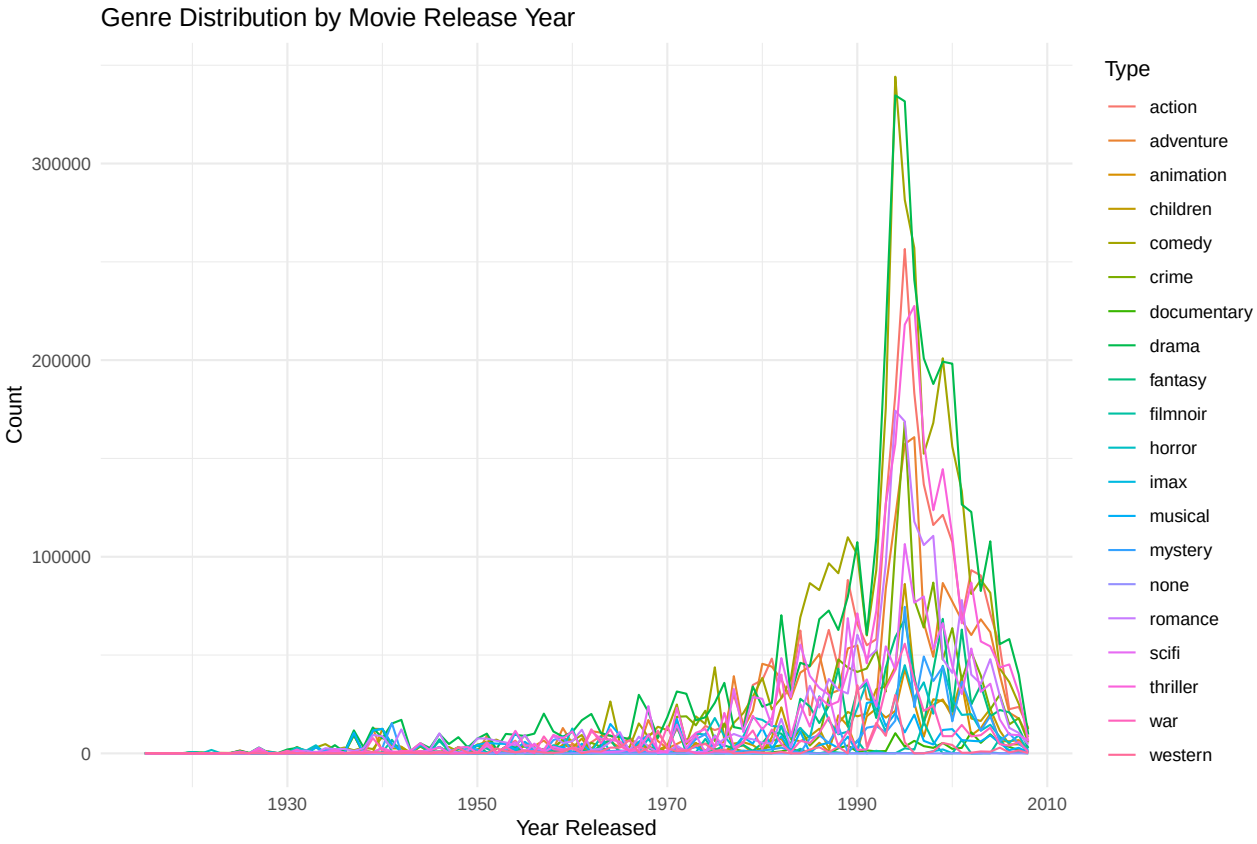


Now we can explore the average rating of each genre, to see which genres tend to be rated higher or lower.

```
## [1] 3.436908
```



Next, we can look at the relationship between movie genres and movie release dates to see genre popularity trends.

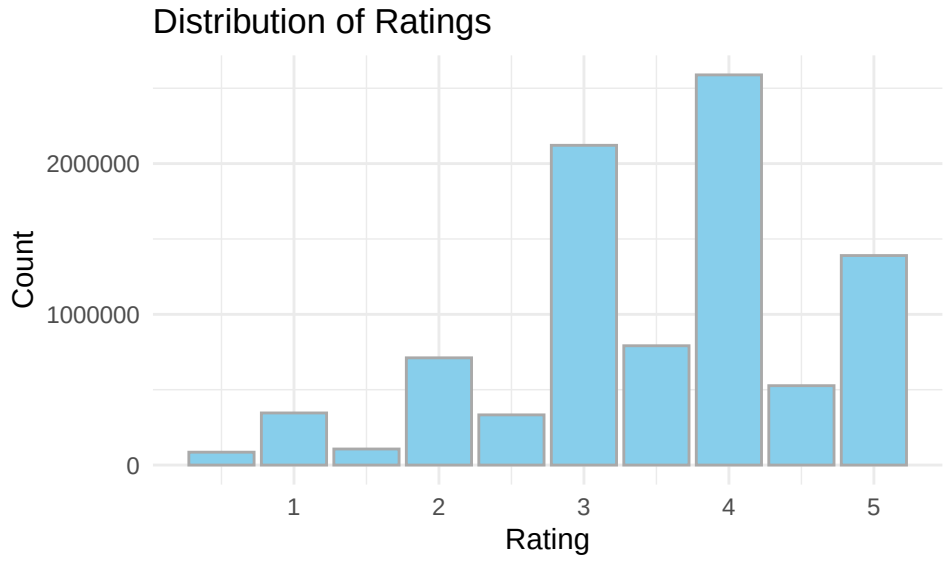


Data Exploration: Rating Values

We can also explore the breakdown of the rating data to see which rating values were most popular. There are 10 different ratings that were given to all the movies, ranging from 1 to 5. Note that there are no 0 star ratings in this dataset, the lowest rating is 0.5 stars. We see that the most common rating that is given is 4 stars and 0.5 stars is the least common.

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

Further, we can examine the data to determine if full star or half star ratings are more common. We can see in general there are fewer half star ratings given than full star ratings.



Modelling

Overview of Machine Learning Techniques Used

This report details the development of 7 different machine learning models, using various teachniques and input variables.

1.Linear Model - Movie Effect

2.Linear Model - User Effect

3.Linear Model - Movie & User Effect

4.Linear Model - Genre Effect

5.Linear Model - Movie, User & Genre Effect

6.Linear Model - Movie, User, Genre & Year Effect

7.Regularized Model - Movie, User, Genre & Year Effects

First, we can create a baseline model using just the rating averages in order to have a starting point to compare our more advanced models against. We will assess this model (and all future models) using the RMSE (root mean square error), in order to quantify how far the predicted values are from the observed values in the final hold out validation set. We want to minimize the RMSE, meaning we want to minimize how far our predicted ratings are from our observed ratings in the validation set.

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

We can create an RMSE function based on the above formula:

```
#Create Root Mean Squared Error (RMSE) function
rmse <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))}
```

Baseline Model: Average Rating

This baseline model uses the average movie rating in the dataset and predicts the same rating regardless of the user. Clearly this will not give us a very accurate prediction since there are many other factors such as genre, movie title etc. that are significant in predicting a users rating of a given movie. This baseline model using the mean rating of the dataset is slightly better than randomly guessing ratings, however we know from our data exploration that we can create much more complex and accurate models. Our baseline model is of the form:

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

with

$\epsilon_{u,i}$: independent errors sampled from the same distribution centered at 0 μ : the "true" rating for all movies.

```
set.seed(1)
#Calculate the average rating of all movies in the dataset
avg_rating <- mean(edx$rating)
#Calculate RMSE using the average rating as the predicted value for each movie title
#and the ratings from the final hold out validation set as the observed values
rmse(validation$rating,avg_rating)
```

```
## [1] 1.061202
```

Model 1: Linear Model - Movie Effect

From our exploratory analysis of the dataset, we know that some movies are simply rated much higher (or conversely much lower) than others. Thus we need to reflect this pattern in our model in order to account for ratings of very popular/unpopular movies.

Now we can expand our baseline model to include a term b_i that represents the movie effect, thus our model becomes:

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

We can use linear regression and least squares estimation in order to estimate our movie effect b_i, however since there are 10,677 distinct movie ID's in our dataset this method will be very time consuming.

```
model1 <- lm(rating ~ as.factor(movieId), data = edx)
```

Alternatively, we can use the following equation derived from our model in order to estimate our movie effect b_i:

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

We can see that using the movieId variable helps to more accurately predict a users rating. As a result, our RMSE decreases slightly from the baseline model, reflecting the improvement in prediction.

```
#Find the movie ID estimates
movie_effects <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - avg_rating))

#Add the b_i estimates to our baseline model
predicted_ratings <- avg_rating + validation %>%
```

```
  left_join(movie_effects, by='movieId') %>%
  pull(b_i)

#Check how our RMSE improves from our baseline model with the movie effect term
rmse(predicted_ratings, validation$rating)
```

```
## [1] 0.9439087
```

Model 2: Linear Model - User Effect

Using our data exploration and our intuition, we know that some users are more active than others in giving ratings. Thus we can create a model to capture the user effect b_u on rating prediction:

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

For the same time concerns expressed with the movie effect model, we will calculate our user effect using the following equation, rather than the lm function:

$$\hat{b_u} = Y_{u,i} - \mu$$

```
model2 <- lm(rating ~ as.factor(userId))
```

The resulting RMSE is an improvement again from our baseline model, demonstrating the significance of the user effect in predicting ratings. However note that the user effect model doesn't perform quite as well as the movie effect model.

```
#Find the user ID estimates
user_effects <- edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - avg_rating))

#Add the b_u estimates to our baseline model
predicted_ratings <- avg_rating + validation %>%
  left_join(user_effects, by='userId') %>%
  pull(b_u)

#Check how our RMSE improves from our baseline model with the user effect term
rmse(predicted_ratings, validation$rating)
```

```
## [1] 0.978336
```

Model 3: Linear Model - Movie & User Effect

Since we know that both the movie and user effects are important in making rating predictions, we can combine them to create a model that captures both the movie and user effects:

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

Using the same methodology as the first two models, instead of using the lm function, we get the following equation:

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

```
model3 <- lm(rating ~ as.factor(movieId) + as.factor(userId))
```

The resulting RMSE from this user & movie effect model is even lower than the baseline and first two models.

```
#Find the user ID estimates given the movie ID effect
user_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - avg_rating - b_i))

#Add the b_i & b_u estimates to our baseline model
predicted_ratings <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  mutate(prediction = avg_rating + b_i + b_u) %>%
  pull(prediction)

#Check how our RMSE improves from our baseline model with the movie & user effect terms
rmse(predicted_ratings, validation$rating)
```

```
## [1] 0.8653488
```

Model 4: Linear Model - Genre Effect

We know that some genres tend to have higher ratings than others and that some genres are rated more than others, so we can look at how the movie genre affects a users rating. We have 19 different genre categories in our dataset, with some movies falling within multiple genre categories, while others are just classified as one genre. We can use our dummy variable genre columns as independent variables and create a linear model predicting rating. We can see from the summary of this model that all of the genres are significant in predicting a users movie rating since they all have p-values significantly lower than our cut off α=0.05. The rmse of this model is lower than our baseline model, indicating that combining these effects with our movie and user effects will be helpful in making accurate rating predictions.

```
#Create a linear model with genre types as predictors
model4 <- lm(rating~comedy+romance+action+crime+thriller+drama+scifi+
               adventure+children+fantasy+war+animation+musical+western+mystery+
               filmnoir+horror+documentary+imax,data = edx)
summary(model4)
```

```
##
## Call:
## lm(formula = rating ~ comedy + romance + action + crime + thriller +
##     drama + scifi + adventure + children + fantasy + war + animation +
##     musical + western + mystery + filmnoir + horror + documentary +
##     imax, data = edx)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8478 -0.6003  0.1694  0.6694  2.0518
##
## Coefficients:
##               Estimate Std. Error  t value             Pr(>|t|)
## (Intercept)  3.3985573  0.0010539 3224.782 < 0.0000000000000002 ***
## comedy      -0.0679405  0.0008755  -77.604 < 0.0000000000000002 ***
```

```
## romance      0.0347627  0.0009393   37.011 < 0.0000000000000002 ***
## action      -0.1337645  0.0009846 -135.853 < 0.0000000000000002 ***
## crime        0.1785096  0.0010815  165.062 < 0.0000000000000002 ***
## thriller     0.0142954  0.0009603   14.886 < 0.0000000000000002 ***
## drama        0.2249380  0.0008658  259.790 < 0.0000000000000002 ***
## scifi       -0.0181263  0.0010896  -16.636 < 0.0000000000000002 ***
## adventure    0.1188469  0.0010160  116.976 < 0.0000000000000002 ***
## children    -0.2486945  0.0017997 -138.190 < 0.0000000000000002 ***
## fantasy      0.0700696  0.0012494   56.082 < 0.0000000000000002 ***
## war          0.2400492  0.0015678  153.113 < 0.0000000000000002 ***
## animation    0.2855228  0.0021072  135.500 < 0.0000000000000002 ***
## musical      0.0843977  0.0017903   47.141 < 0.0000000000000002 ***
## western      0.0649046  0.0024498   26.494 < 0.0000000000000002 ***
## mystery      0.1047972  0.0015229   68.814 < 0.0000000000000002 ***
## filmnoir     0.3362027  0.0031470  106.833 < 0.0000000000000002 ***
## horror      -0.1808623  0.0014099 -128.283 < 0.0000000000000002 ***
## documentary  0.3764561  0.0035504  106.033 < 0.0000000000000002 ***
## imax         0.0918325  0.0115593    7.945  0.00000000000000195 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.041 on 9000035 degrees of freedom
## Multiple R-squared:  0.0357, Adjusted R-squared:  0.0357
## F-statistic: 1.754e+04 on 19 and 9000035 DF,  p-value: < 0.00000000000000022
```

```
predicted_ratings <- predict(model4,validation)
rmse(predicted_ratings, validation$rating)
```

```
## [1] 1.042038
```

Model 5: Linear Model - Movie, User & Genre Effect

Now using the findings from our analysis, we can combine models 3 & 4 to get an even more accurate model that inclues movie title, user and genre effects. The RMSE is minimized even further when we account for the genres of the movies.

```
predicted_ratings <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(comedy_effects, by='comedy') %>%
  left_join(drama_effects, by='drama') %>%
  left_join(action_effects, by='action') %>%
  left_join(thriller_effects, by='thriller') %>%
  left_join(musical_effects, by='musical') %>%
  left_join(filmnoir_effects, by='filmnoir') %>%
  left_join(romance_effects, by='romance') %>%
  left_join(crime_effects, by='crime') %>%
  left_join(scifi_effects, by='scifi') %>%
  left_join(adventure_effects, by='adventure') %>%
  left_join(children_effects, by='children') %>%
  left_join(fantasy_effects, by='fantasy') %>%
  left_join(war_effects, by='war') %>%
  left_join(animation_effects, by='animation') %>%
  left_join(western_effects, by='western') %>%
```

```
  left_join(mystery_effects, by='mystery') %>%
  left_join(horror_effects, by='horror') %>%
  left_join(documentary_effects, by='documentary') %>%
  left_join(imax_effects, by='imax') %>%
  mutate(prediction = avg_rating+b_i+b_u+b_c+b_d+b_a+b_t+b_m+b_n+b_r+b_cr+b_s
                      +b_ad+b_ch+b_f+b_w+b_an+b_we+b_my+b_h+b_do) %>%
  pull(prediction)

#Check how our RMSE improves from our baseline model with the movie,user & genre effect terms
rmse(predicted_ratings, validation$rating)
```

```
## [1] 0.8652027
```

Model 6: Linear Model - Movie, User, Genre & Year Effects

To further improve upon our model containing movie, user and genre effects we can add term to represent the year effect, i.e. the year a movie was released. We know that the age of a movie is related to the amount of ratings a movie has in our dataset from the distribution observed in the visualization section. Since movies released between 1993 and 2000 have more ratings than movies released between 1915-1970 and 2003-2008, an effect term to capture this trend will improve our prediction.We can see that the rating predictions using this model further reduced the RMSE from our previous models.

```
predicted_ratings <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(comedy_effects, by='comedy') %>%
  left_join(drama_effects, by='drama') %>%
  left_join(action_effects, by='action') %>%
  left_join(thriller_effects, by='thriller') %>%
  left_join(musical_effects, by='musical') %>%
  left_join(filmnoir_effects, by='filmnoir') %>%
  left_join(romance_effects, by='romance') %>%
  left_join(crime_effects, by='crime') %>%
  left_join(scifi_effects, by='scifi') %>%
  left_join(adventure_effects, by='adventure') %>%
  left_join(children_effects, by='children') %>%
  left_join(fantasy_effects, by='fantasy') %>%
  left_join(war_effects, by='war') %>%
  left_join(animation_effects, by='animation') %>%
  left_join(western_effects, by='western') %>%
  left_join(mystery_effects, by='mystery') %>%
  left_join(horror_effects, by='horror') %>%
  left_join(documentary_effects, by='documentary') %>%
  left_join(imax_effects, by='imax') %>%
  left_join(year_effects, by='year_released') %>%
  mutate(prediction = avg_rating+b_i+b_u+b_c+b_d+b_a+b_t+b_m+b_n+b_r+b_cr+b_s
                      +b_ad+b_ch+b_f+b_w+b_an+b_we+b_my+b_h+b_do+b_y) %>%
  pull(prediction)

#Check how our RMSE improves from our baseline model with the movie,user,genre & year effect terms
rmse(predicted_ratings, validation$rating)
```

```
## [1] 0.8648744
```

Model 7: Regularized Model - Movie, User, Genre & Year Effects

Regularization allows us to penalize large estimates that are formed using small sample sizes, eliminating a lot of variability. In the above models some of our estimates (such as $\hat{b_i}$ and $\hat{b_u}$) have high variability because a user only rated a small number of movies, or a movie was only rated a few times. Since we have a small number of of ratings in these two cases, there is more variability in our prediction and a penalty term can help to minimize this bias. By constraining the total variability of the effect sizes with penalized regression (penalized least squares) the equation that we want to minimize becomes:

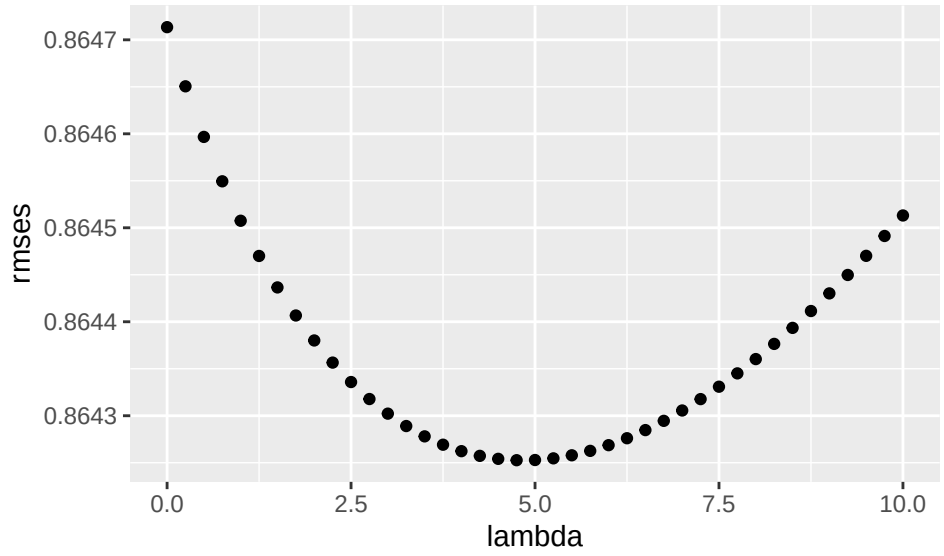
$$\hat{b_i} = \Sigma(Y_{u,i} - \hat{\mu})/(\lambda + n_i)$$

We want to find the value of the penalty factor lambda (tuning parameter) that will minimize the RMSE. This will make the effects with a smaller number of ratings less significant in the prediction.

Note that this model doesn't compute each individual genre effect for each of the 19 genres, but rather uses the 797 genre groupings in the genres column since the computation for each genre type and for multiple lambda values would be too lengthy. We can see that using this regularized model with the optimal tuning parameter lambda, we further reduce the RMSE.

```
#Find the optimal lambda by first testing a sequence of potential lambdas
lambda <- seq(0, 10, 0.25)
#Calculate the rmses for each of the lambda values
rmses <- sapply(lambda, function(l){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - avg_rating)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - avg_rating)/(n()+l))

  b_y <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year_released) %>%
    summarize(b_y = sum(rating-b_i-b_u-avg_rating)/(n()+l))

  b_g <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by='year_released') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating-b_i-b_u-b_y-avg_rating)/(n()+l))

  predicted_ratings <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_y, by = "year_released") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = avg_rating + b_i + b_u + b_y + b_g) %>%
    pull(pred)

    return(RMSE(predicted_ratings, validation$rating))
```

```
})
qplot(lambda, rmses)
```




```
#The optimal lambda is 4.75
min_lambda <- lambda[which.min(rmses)]
min(rmses)
```

```
## [1] 0.8642527
```

Results

We can see from the below table of RMSE for each of the 7 models that the regularized model performed best in comparison to the rest. In general, as we added each of the different effects RMSE was continuously minimized. Adding each of the movie, user, genre and year effects to the model as well as accounting for the outliers in the dataset using penalized least squares estimation in model 7 ultimately produced the most accurate predictions.

Conclusion

This report successfully trained and built 7 different machine learning algorithms to predict a user's movie rating based on the attributes in the MovieLens dataset. Using the cleaned edx dataset as our training set and a final hold out validation set as our test set, we were able to successfully construct a regularized model with a low RMSE of 0.8642527. Overall the regularized model, which includes the movie title, user, genre and year effects performed the best (resulted in the smallest RMSE) of all the models built and is a successful recommendation system.

Limitations & Future Work

Future work on this project would include the further exploration of ensemble methods and other machine learning models that could potentially improve the RMSE/prediction results. However computer memory and time limitations are a constraint to consider.