MovieLens Project

Peter Mulhall

10/29/2020

1.0 Introduction

1.1 Dataset Description

MovieLens is a movie recommendation website run by GroupLens, a research lab out of the University of Minnesota. They have offered a data set of 10M movie reviews that we will utilize for this analysis.

Below is the head and summary of the dataset for familiarization:

```
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:          Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

1.2 Project Goals

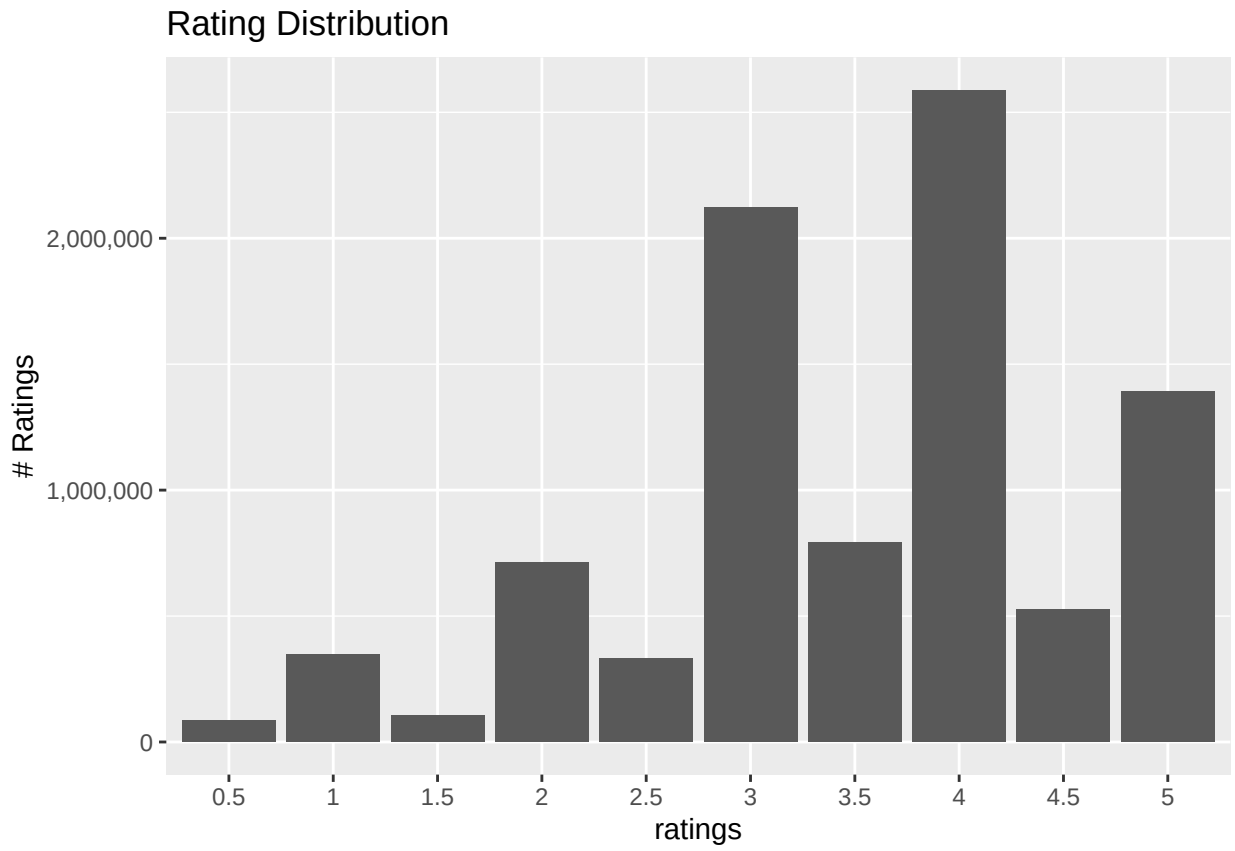
2.0 Methods and Analysis

2.1 Data Cleaning

As the dataset was compiled by GroupLens, it was largely clean and usable as-is. The only significant cleaning tasks were to separate out the year from the Title, and to split movies with multiple genres into new rows so that the data could be used in analysis.

2.2 Data Exploration and Visualization

```
ratings <- as.vector(edx$rating)
ratings <- ratings[ratings != 0]
ratings <- factor(ratings)
qplot(ratings) +
  ggtitle("Rating Distribution") +
  ylab("# Ratings") +
  scale_y_continuous(name = "# Ratings", labels = comma)
```



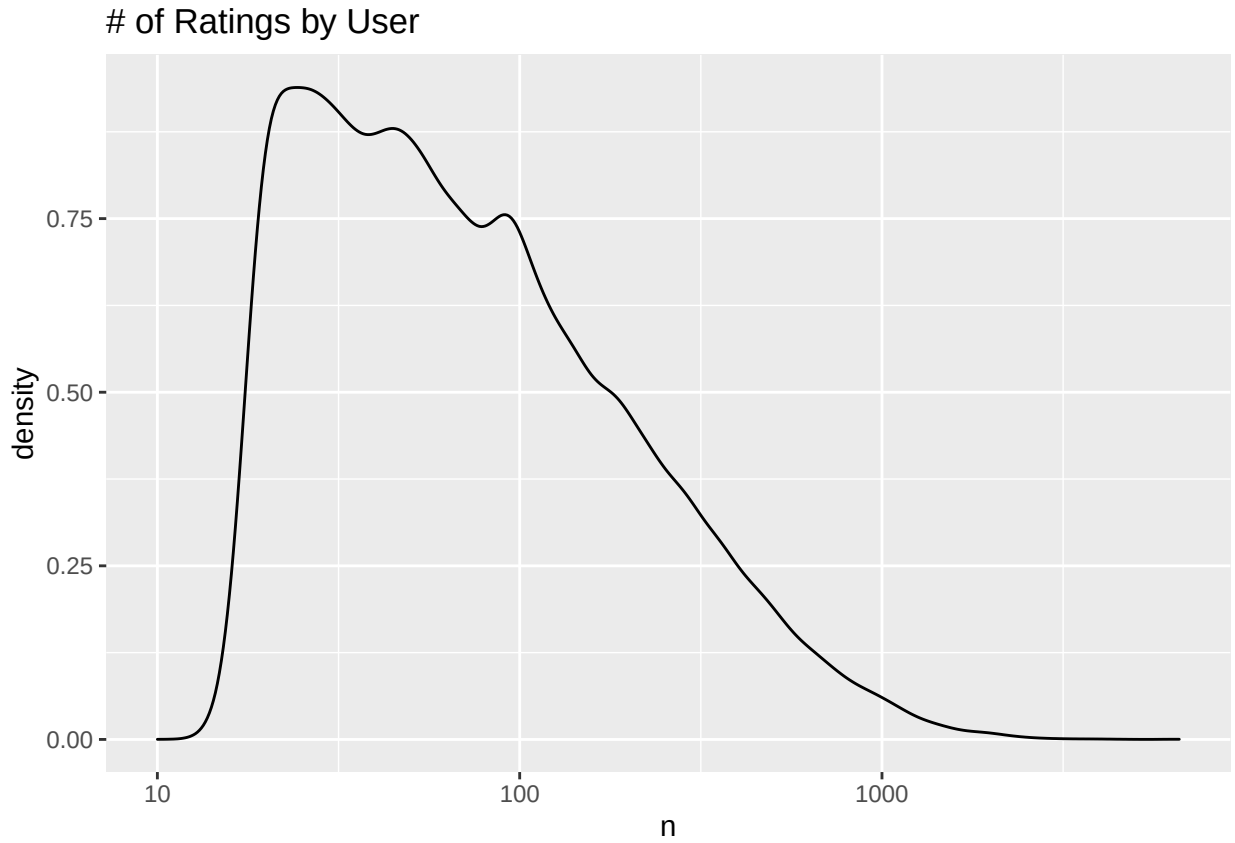
We can see that users prefer to rate on whole star ratings much more frequently than half star ratings.

```
edx %>% summarise(
  unique_movies = n_distinct(movieId),
  unique_users = n_distinct(userId),
  unique_genres = n_distinct(genres),
  unique_years = n_distinct(year)
)
```

```
##   unique_movies unique_users unique_genres unique_years
## 1         10677        69878           797           94
```

Next we can take a look at the distribution of user's ratings. From the graph below it is easy to see that the most common ratings are 4 and 3, and that users actively avoid half star ratings.

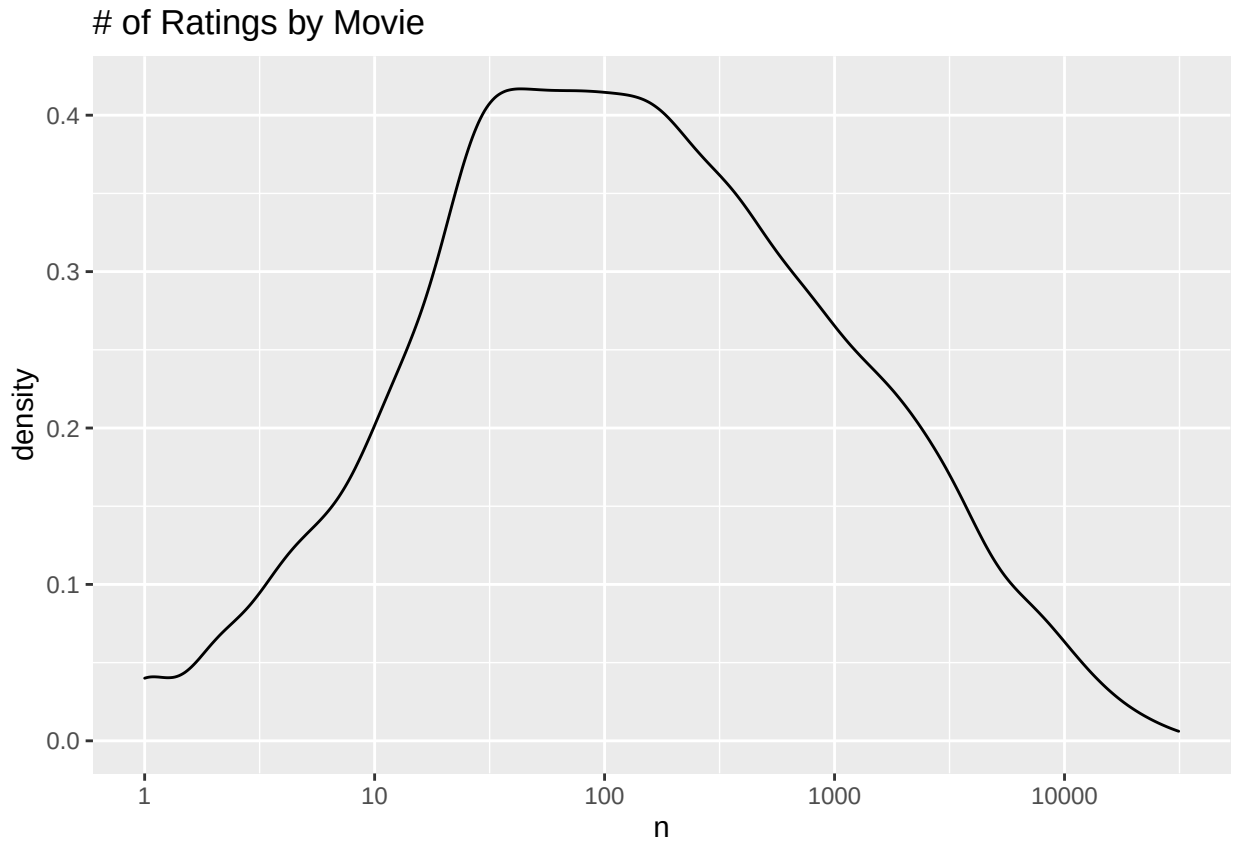
```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_density() +
  scale_x_log10() +
  ggtitle("# of Ratings by User")
```



Some users rate hardly any movies, while others rate thousands - we should consider this when evaluating the ratings for prediction purposes.

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
```

```
  geom_density() +
  scale_x_log10() +
  ggtitle("# of Ratings by Movie")
```



Like the user rating data, some movies have very few ratings, while others have more than 10,000 ratings within our database sample.

2.3 Modelling Approach

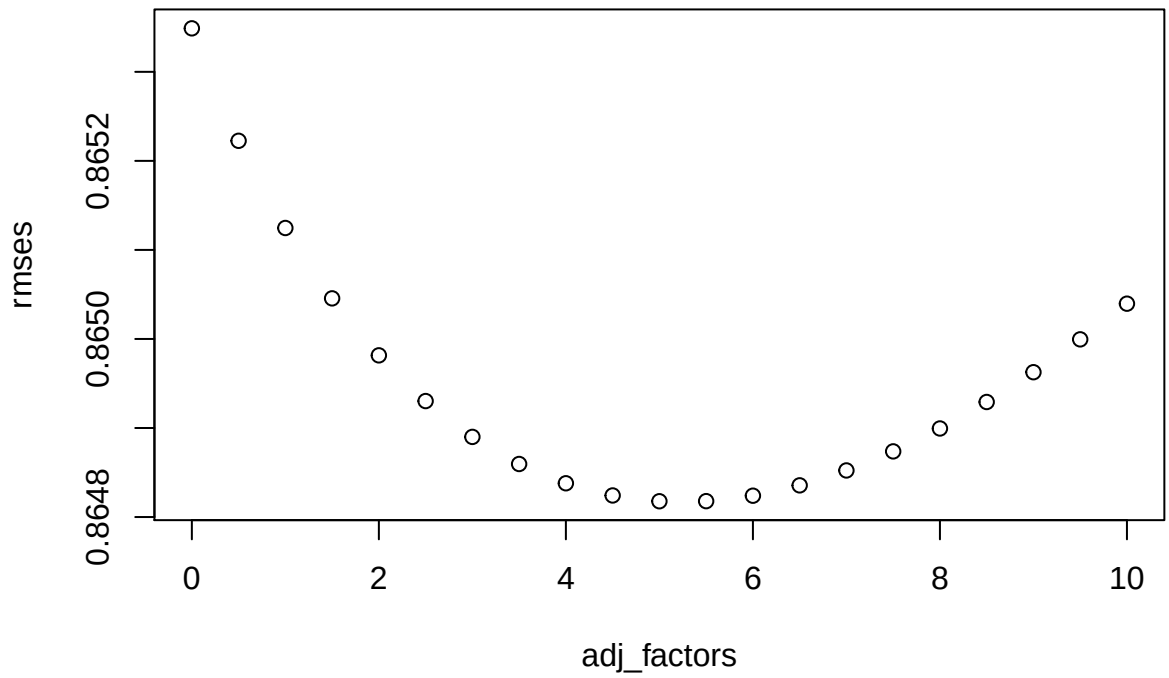
3.0 Results

Achieved an RMSE < 0.86490 by predicted ratings based on adjusting the average rating overall in combination with the particular movie's rating and the user's rating history. Lastly, movies with low numbers of ratings were penalized after calculating an adjustment factor.

3.1 Modelling Results

```
# Root Mean Square Error Function
RMSE <- function(actual_rating, predicted_rating)
{
  sqrt(mean((actual_rating - predicted_rating)^2))
}
```

```r
adj_factors <- seq(0, 10, 0.5) #test for lambda value
rmses <- sapply(adj_factors, function(l){

  mts <- mean(edx$rating) # mean rating of training set

  me <- edx %>%
    group_by(movieId) %>% #adjust by movie rating
    summarize(me = sum(rating - mts)/(n()+l), .groups = 'drop') # penalize low number of ratings

  # adjust mean by user ratings and movie ratings and penalize low number of ratings
  am <- edx %>%
    left_join(me, by="movieId") %>%
    group_by(userId) %>% #adjust by user rating
    summarize(am = sum(rating - me - mts)/(n()+l), .groups = 'drop')

  # calculate predicated ratings based on movie and user effects
  predicted_ratings <-
    validation %>%
    left_join(me, by = "movieId") %>%
    left_join(am, by = "userId") %>%
    mutate(pred = mts + me + am) %>% # combine all three adjustments to make a prediction
    .$pred

  return(RMSE(predicted_ratings, validation$rating))
})
plot(adj_factors, rmses)
```




```
adj_factor <- adj_factors[which.min(rmses)]
paste('RMSE:',min(rmses))
```

```
## [1] "RMSE: 0.864817746466669"
```

4.0 Conclusion

In conclusion, the MovieLens data was an interesting look at using machine learning to predict ratings that may be used to predict user preferences.

The fact that the model was able to calculate ratings within one star on average without any idea to a user's taste was quite impressive.

There are many data points that can be used to further progress these models, such as considering the ratings of specific genres by user, or preferences by year the movie was made.