Movie Recommendation System

Lori Stevens

5/11/2020

1 Introduction

Recommendation systems are designed to understand and predict user preferences based on observed behavior. This is a critical form of machine learning in Retail and e-Commerce industries to personalize interactions with customers. Many consumers interact with recommendation systems daily, receiving suggestions for products or movies of interest on websites and streaming services.

This report details the creation of a movie recommendation system using the MovieLens dataset collected by GroupLens research. The objective is to train a machine learning algorithm to predict moving ratings utilizing a subset of the 10M version of the MovieLens dataset. This is a collection of 10M ratings released in January 2009. The full dataset is available at the following location:

https://grouplens.org/datasets/movielens/latest/ (https://grouplens.org/datasets/movielens/latest/)

Four models are created and compared in this report. The root mean squared error (RMSE) is used to evaluate and identify the best performing algorithm. The RMSE is a measure of the algorithm's error in the prediction of user ratings. The goal is to have an RMSE significantly lower than the baseline RMSE to be calculated in the modeling process. All code is referenced directly in this report to allow for easy review of approach, findings, and code.

2 Analysis

This section explains the process and techniques used, including data creation, exploration, visualization, modeling approach, and insights.

2.1 Data Creation

The first step is to create the edx and validation datasets. The edx dataset is further split into a train and test dataset, to train and test the models. The validation dataset will be used to evaluate the performance of the final algorithm.

```r
# Install packages
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-projec
  t.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-proj
  ect.org")

# Load libraries
library(dplyr)
library(tidyverse)
library(tidyr)
library(stringr)
library(ggplot2)
library(caret)
library(lubridate)

# Download the 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.da
  t"))),
                 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")
movies <- as.data.frame(movies) %>%
  mutate(movieId = as.numeric(levels(movieId))[movieId],
  title = as.character(title),
  genres = as.character(genres))

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

# Create validation dataset (10% of MovieLens dataset)
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FAL
  SE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Validate 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)

# Remove unnecessary data
rm(dl, ratings, movies, test_index, temp, movielens, removed)

# Convert timestamp to YYYY-MM-DD HH:MM:SS format
edx$timestamp <- as.POSIXct(edx$timestamp, origin="1970-01-01")
validation$timestamp <- as.POSIXct(validation$timestamp, origin="1970-01-01")

# Create year column in datasets
edx$year <- format(edx$timestamp,"%Y")
validation$year <- format(validation$timestamp,"%Y")

# Create test dataset (20% of the edx dataset)
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
test_set <- edx[test_index,]

# Create train dataset (80% of the edx dataset)
train_set <- edx[-test_index,]

# Validate userId and movieId in test set are also in train set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

2.2 Data Exploration and Visualization

The dataset includes six variables including the User ID, Movie ID, Genre, Title, Rating, and Timestamp. Each row is representative of a movie rating by a single user. The genre assigned can be inclusive of a combination of defined genres. An example of the data included is as follows:

```
# Generate sample records of edx dataset
head(edx) %>%
  knitr::kable()
```

	userId	movieId	rating	timestamp	title	genres	year
1	1	122	5	1996-08-02 07:24:06	Boomerang (1992)	Comedy|Romance	1996
2	1	185	5	1996-08-02 06:58:45	Net, The (1995)	Action|Crime|Thriller	1996

	userId	movieId	rating	timestamp	title	genres	year
4	1	292	5	1996-08-02 06:57:01	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
5	1	316	5	1996-08-02 06:56:32	Stargate (1994)	Action|Adventure|Sci-Fi	1996
6	1	329	5	1996-08-02 06:56:32	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996
7	1	355	5	1996-08-02 07:14:34	Flintstones, The (1994)	Children|Comedy|Fantasy	1996

A summary of the edx dataset validates that there are no missing values. Ratings range on a scale from 0.5 to 5.0.

```
# Create summary of edx dataset
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :1995-01-09 06:46:49
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:2000-01-01 18:11:23
##  Median :35738   Median : 1834   Median :4.000   Median :2002-10-24 17:11:58
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :2002-09-21 09:45:07
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:2005-09-14 22:21:21
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :2009-01-05 00:02:16
##     title              genres              year
##  Length:9000055     Length:9000055     Length:9000055
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

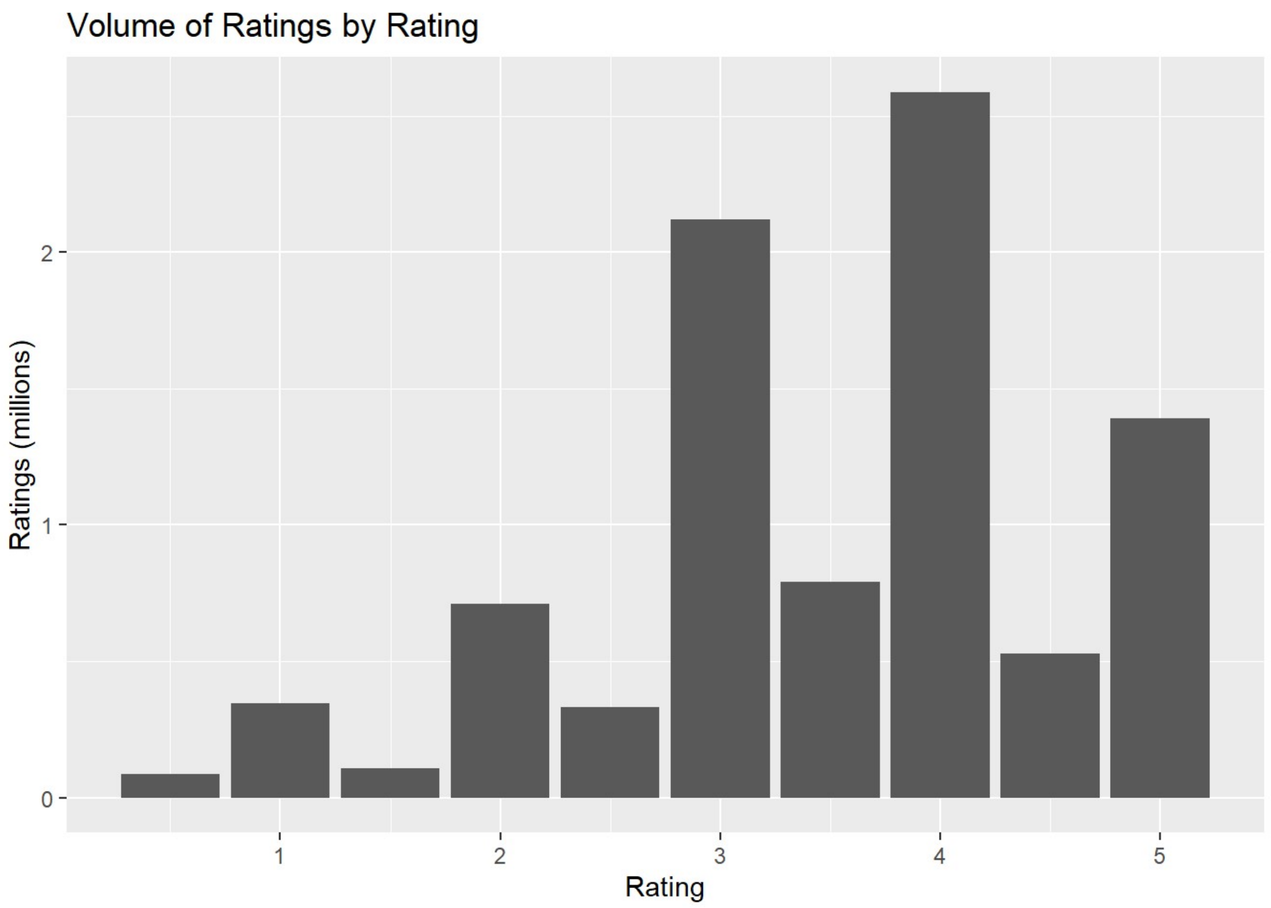
The edx dataset is inclusive of over 10K unique movies by approximately 70K users.

```
# Calculate total number of unique users and movies in edx dataset
edx %>%
  summarize(unique_users = n_distinct(userId), unique_movies = n_distinct(movieId)) %
  >%
   knitr::kable()
```

unique_users	unique_movies
69878	10677

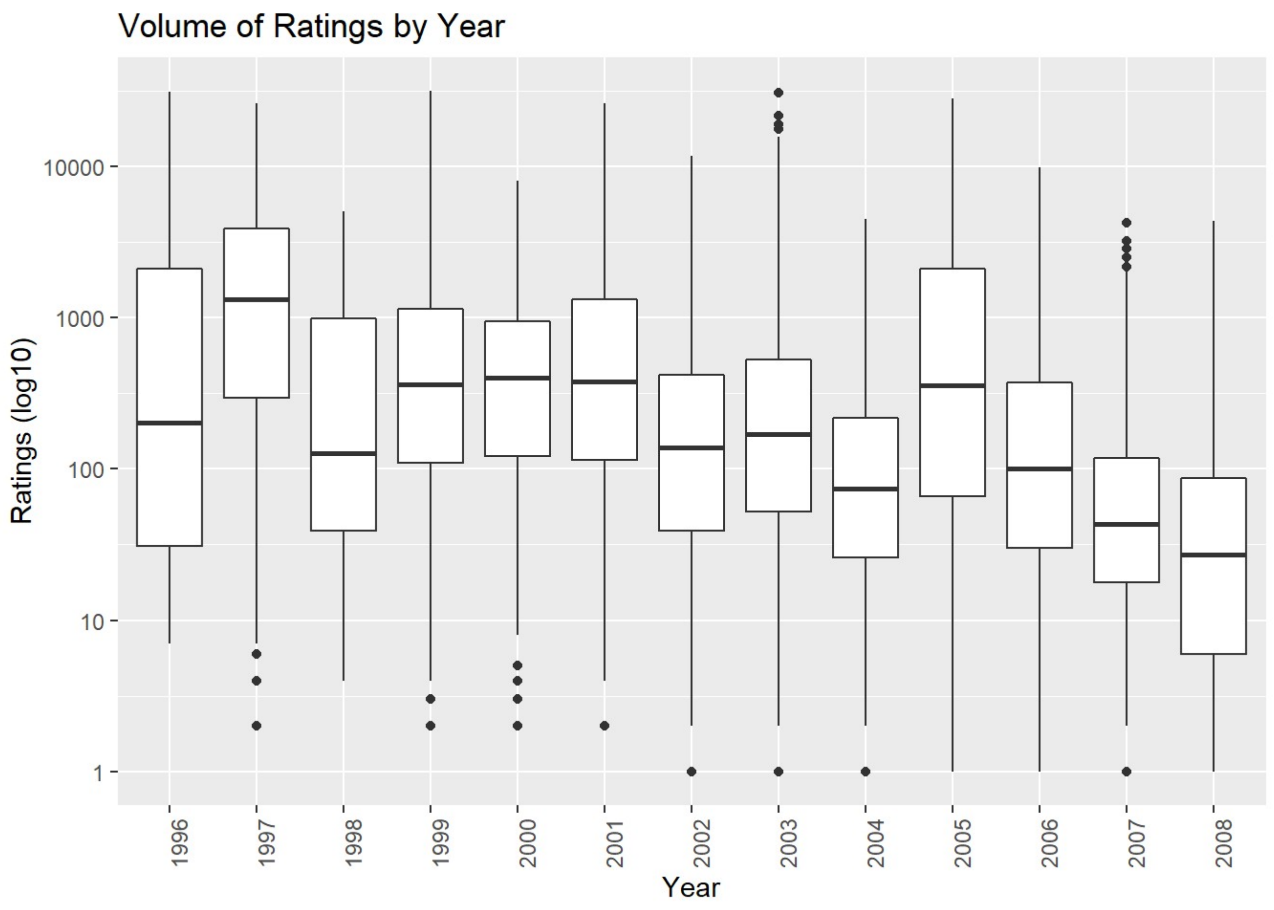
Users have a tendency to rate movies higher overall, with the most frequent rating of 4.0. In addition, users have a tendency to round ratings as half ratings are far less common than full star ratings.

```r
# Create bar chart by rating in edx dataset
edx %>%
  group_by(rating) %>%
  summarize(count = n(), millions = count/10^6) %>%
  ggplot(aes(rating, millions)) +
  geom_bar(stat="identity", show.legend = FALSE) +
  xlab("Rating") +
  ylab("Ratings (millions)") +
  ggtitle("Volume of Ratings by Rating")
```



The median number of ratings per movie has been declining over the years included in the dataset. 2009 was excluded from the chart below as only 5 days of data was available.

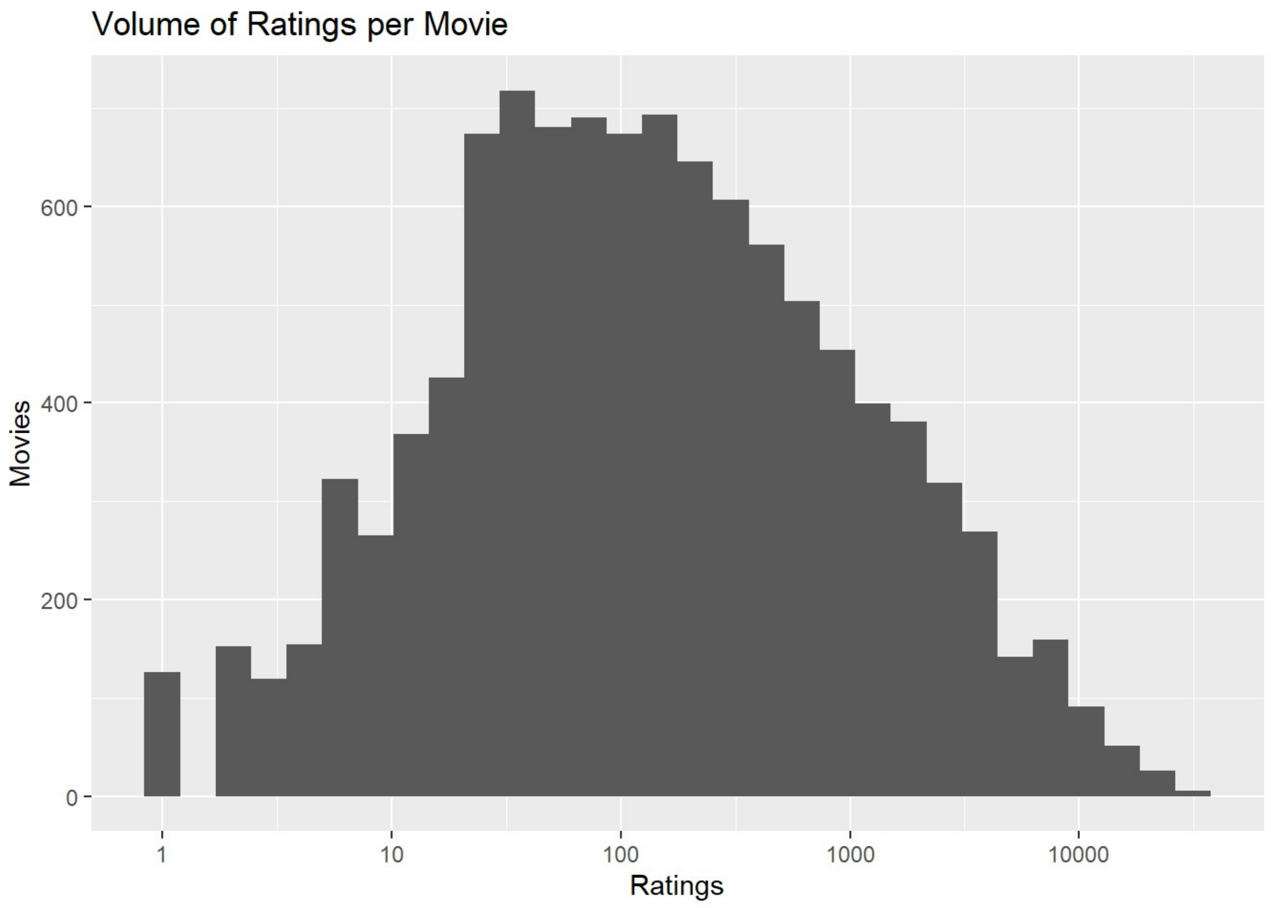
```
# Create boxplot of movie ratings by year
edx %>%
  group_by(movieId) %>%
  filter(year != "2009") %>%
  summarize(n = n(), year = as.character(first(year))) %>%
  qplot(year, n, data = ., geom = "boxplot") +
  scale_y_log10() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Year") +
  ylab("Ratings (log10)") +
  ggtitle("Volume of Ratings by Year")
```



The volume of ratings each movie has received varies significantly, with select movies having a very large or small number of ratings associated. It can be seen in the chart below that there are a particularly large volume of outliers with a single rating.

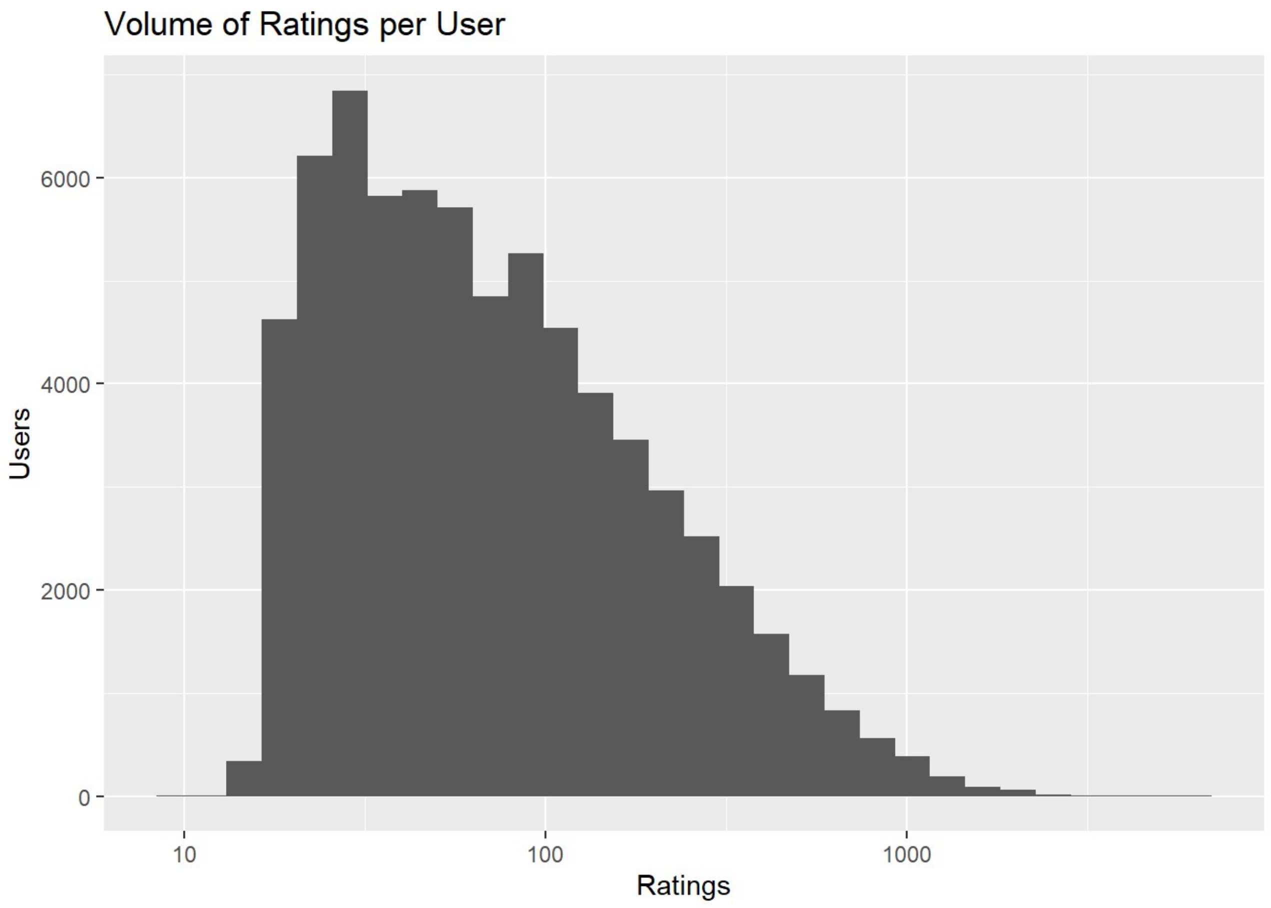
To ensure that these outliers do not impact our model, **regularizaton** is used in the models to include a penalty term for small sample sizes. This is outlined in more detail in the **Data Modeling** section.

```
# Create histogram of volume of ratings per movie
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram() +
  scale_x_log10() +
  xlab("Ratings") +
  ylab("Movies") +
  ggtitle("Volume of Ratings per Movie")
```



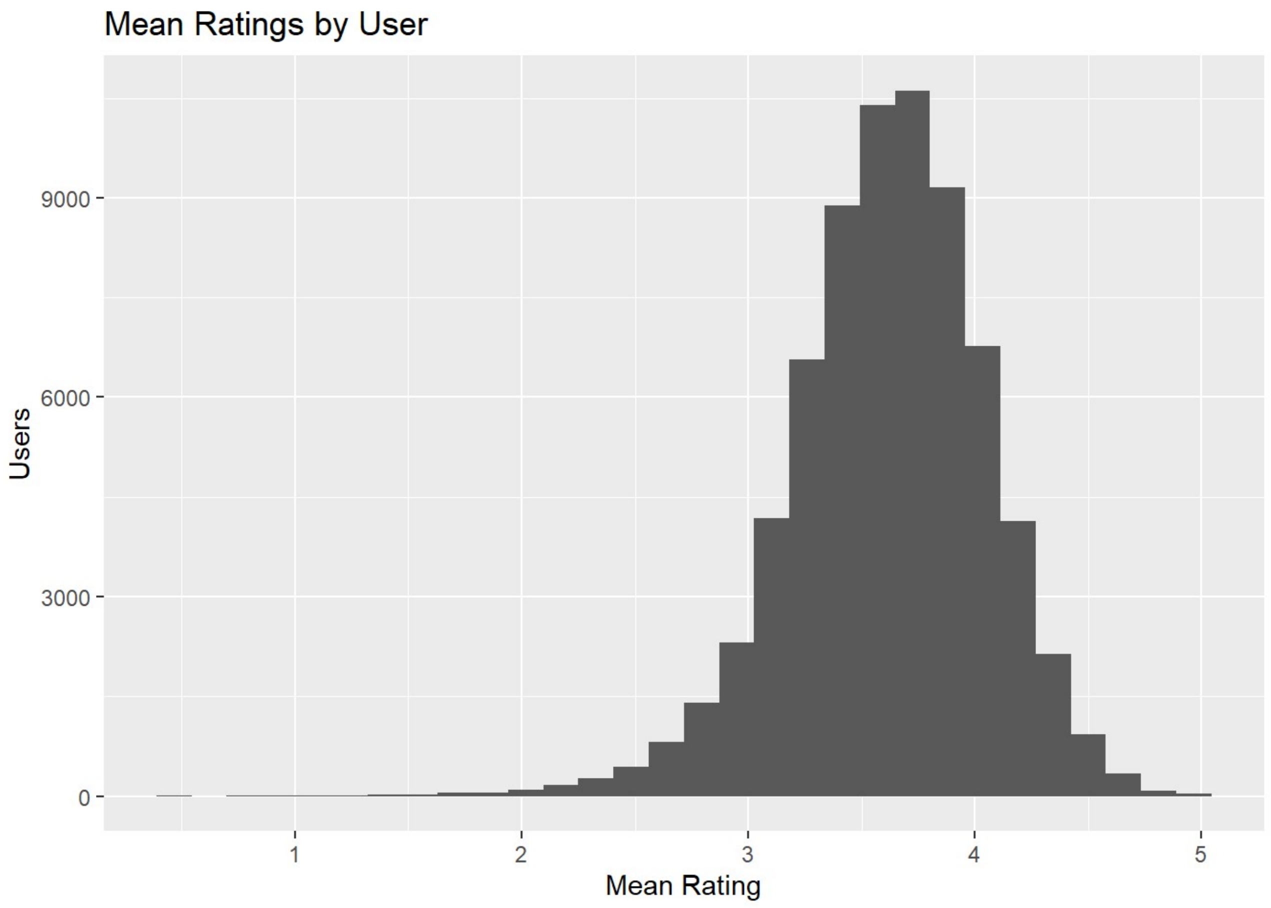
It can be seen in the chart below that many users have rated a large volume of movies. To account for this, a penalty term for users is incorporated into the model as well.

```r
# Create histogram of volume of ratings per user
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram() +
  scale_x_log10() +
  xlab("Ratings") +
  ylab("Users") +
  ggtitle("Volume of Ratings per User")
```



In addition, users vary significantly on how critical they are with their ratings. Select users have a tendency to give lower or higher than average star ratings. A penalty term for highly critical users is also incorporated into the model to account for this.

```
# Create histogram of mean ratings by user
edx %>%
  group_by(userId) %>%
  summarize(mean_rating = mean(rating)) %>%
  ggplot(aes(mean_rating)) +
  geom_histogram() +
  xlab("Mean Rating") +
  ylab("Users") +
  ggtitle("Mean Ratings by User")
```



2.3 Data Modeling

The **root mean squared error (RMSE)** is used to evaluate and identify the best performing algorithm. The RMSE is a measure of the algorithm's error in the prediction of user ratings. The function that computes the RMSE is as follows:

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

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

A successful model will have an RMSE significantly lower than the baseline RMSE. The baseline RMSE is calculated by a basic model that assumes the same rating for all movies and users. The additional models incorporate penalty terms for small sample sizes and highly active or critical users.

2.3.1 Baseline Model

The baseline model is simple model that makes the assumption that all differences in movie ratings are explained by random variation. μ represents the true rating for all movies and users and ϵ represents independent errors sampled from the same distribution centered at zero.

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

In this case, the least squares estimate of μ is the average rating of all movies across all users. The least squares estimate of μ is the estimate that minimizes the root mean squared error. This is calculated as the average of all ratings. The average rating is approximately 3.5 and the baseline RMSE is **1.0599**. This will be used for comparison purposes only.

```
# Calculate average of train dataset
mu_hat <- mean(train_set$rating)
print(mu_hat)
```

```
## [1] 3.512482
```

```
# Calculate RMSE of baseline model
naive_rmse <- RMSE(test_set$rating, mu_hat)
print(naive_rmse)
```

```
## [1] 1.059904
```

```
# Baseline model RMSE results table
rmse_results <- data_frame(Method = "Baseline Model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline Model	1.059904

2.3.2 Movie Effect Model

To improve our model, we will incorporate a penalty term, b_i , that represents the average rating for each movie, i . This will take into account the bias, b , that popular movies are generally rated higher, and vice versa for unpopular movies.

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

In this case, b_i is the average of $Y_{u,i}$ minus the overall mean for each movie i . In this model, we incorporate the penalty term for the movie effect which improves the RMSE to **0.9437**.

```
# Calculate average of train dataset
mu <- mean(train_set$rating)
print(mu)
```

```
## [1] 3.512482
```

```
# Calculate bias of each movie in train dataset
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Predict ratings with movie effect model
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by = 'movieId') %>%
  .$b_i

# Calculate RMSE of movie effect model
model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
print(model_1_rmse)
```

```
## [1] 0.9437429
```

```
# Add movie effect model RMSE results to table
rmse_results <- bind_rows(rmse_results, data_frame(Method = "Movie Effect Model", RMSE
  = model_1_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline Model	1.0599043
Movie Effect Model	0.9437429

2.3.3 User Effect Model

To further improve our model, we will incorporate an additional penalty term, b_u , that represents the average rating for each user, u . This will take into account the bias, b , that some users have a tendency to either be highly critical or overly positive in their ratings of movies. This also includes the movie effect from the previous model, b_i .

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

In this case, b_u is the average of $Y_{u,i}$ minus the overall mean for each user u . In this model, we incorporate the penalty term for the movie effect and the user effect which improves the RMSE to **0.8659**.

```
# Calculate bias of each user in train dataset
user_avgs <- train_set %>%
  left_join(movie_avgs, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Predict ratings with user effect model
predicted_ratings <- test_set %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

# Calculate RMSE of user effect model
model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
print(model_2_rmse)
```

```
## [1] 0.865932
```

```
# Add user effect model RMSE results to table
rmse_results <- bind_rows(rmse_results, data_frame(Method = "User Effect Model", RMSE
  = model_2_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline Model	1.0599043
Movie Effect Model	0.9437429
User Effect Model	0.8659320

2.3.4 Regularization Model

The final improvement our model uses **regularlization**. This will penalize large estimates that come from small sample sizes. This will take into account that some movies have very few ratings and some users only rated a small number of movies. To ensure that these small sample sizes do not influence the prediction, the tuning parameter lambda, λ , is identified through this process and incorporated into the model. This model also includes the movie effect and user effect from the previous model, b_i and b_u respectively.

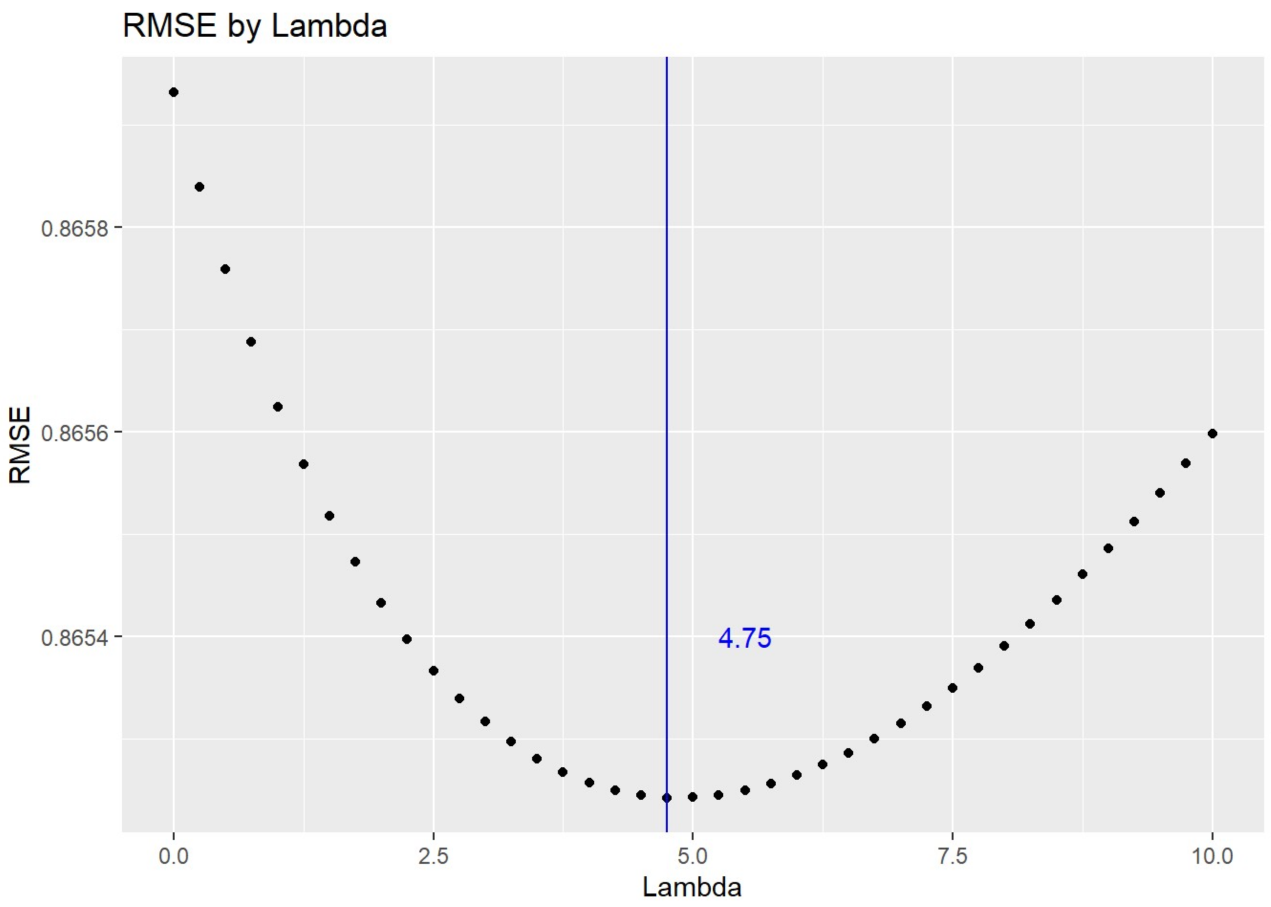
$$\frac{1}{N} + \sum_{u,i}(y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$$

In this case, λ is best tuning parameter identified in the cross validation process. The λ selected is **4.75**. In this model, we incorporate regularization to the penalty terms for the movie effect and the user effect which improves the RMSE to **0.8652**.

```
# Create sequence of potential lambdas to evaluate
lambdas <- seq(0, 10, 0.25)

# Use cross validation to identify ideal lambda as a tuning parameter
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <- test_set %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))})

# Plot lambas for evaluation
qplot(lambdas, rmses) +
  geom_vline(xintercept=4.75, col = "blue") +
  geom_text(data = data.frame(x=5.5, y=0.8654), mapping = aes(x, y, label="4.75"), col
or="blue") +
  xlab("Lambda") +
  ylab("RMSE") +
  ggtitle("RMSE by Lambda")
```



```
# Save optimal lambda for use in model
lambda <- lambdas[which.min(rmses)]
```

```
# Caculate regularized estimate of b_i
b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))

# Calculate regularized estimate of b_u
b_u <- train_set %>%
    left_join(b_i, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Predict ratings with regularization model
predicted_ratings <- test_set %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

# Calculate RMSE of regularization model
model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
print(model_3_rmse)
```

```
## [1] 0.8652421
```

```
# Add regularization model RMSE results to table
rmse_results <- bind_rows(rmse_results, data_frame(Method = "Regularization Model", RM
  SE = model_3_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline Model	1.0599043
Movie Effect Model	0.9437429
User Effect Model	0.8659320
Regularization Model	0.8652421

3 Results

The final step is evaluate the final model with the validation dataset. In prior steps, the train and test dataset have been utilized to identify the best model. The validation dataset is then used to evaluate the performance of the final algorithm. The final RMSE is **0.8648**.

```r
# Caculate regularized estimate of b_i
b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))

# Calculate regularized estimate of b_u
b_u <- edx %>%
    left_join(b_i, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Predict ratings with regularization model
predicted_ratings <- validation %>%
    left_join(b_i, by = 'movieId') %>%
    left_join(b_u, by = 'userId') %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

# Calculate RMSE of regularization model
final_rmse <- RMSE(predicted_ratings, validation$rating)
print(final_rmse)
```

```
## [1] 0.8648201
```

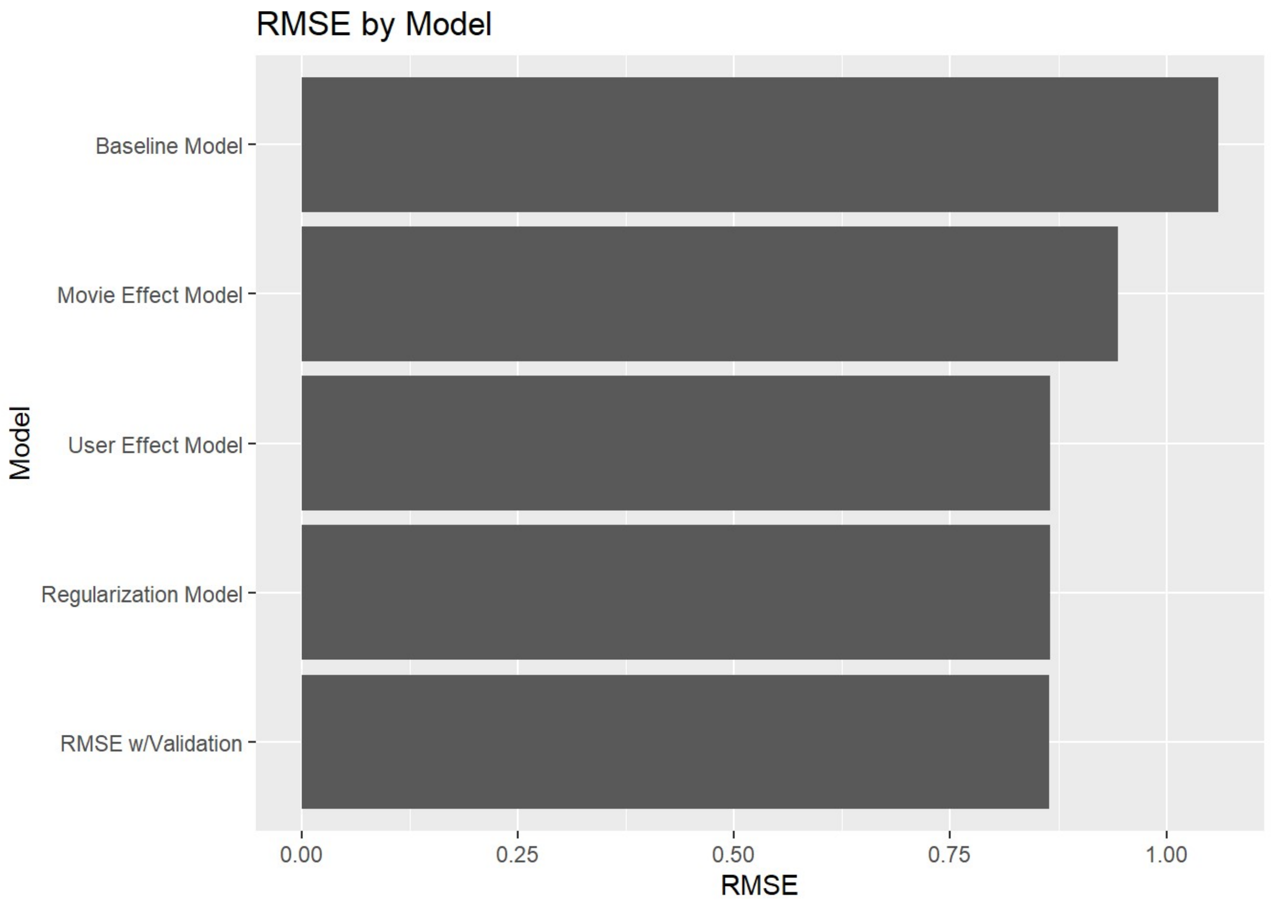
```r
# Add final calculation of RMSE results to table
rmse_results <- bind_rows(rmse_results, data_frame(Method = "RMSE w/Validation", RMSE
  = final_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Baseline Model	1.0599043
Movie Effect Model	0.9437429
User Effect Model	0.8659320
Regularization Model	0.8652421
RMSE w/Validation	0.8648201

4 Conclusion

By taking into account movie and user bias as well as regularization to address sample size variation, a machine learning algorithm to predict ratings and recommend movies was improved significantly. The model that included all elements incorporated has a low RMSE of **0.848**.

```
# Create bar chart by model
rmse_results %>%
  mutate(Method = reorder(Method, RMSE)) %>%
  ggplot(aes(Method, RMSE)) +
  geom_bar(stat="identity", show.legend = FALSE) +
  coord_flip() +
  xlab("Model") +
  ylab("RMSE") +
  ggtitle("RMSE by Model")
```



The RMSE may be further improved through evaluation and incorporation of other potential effects. Examples could include but are not limited to genre preferences and trends in ratings over time. The possibilities are endless, however, limited in this case to the data available within the MovieLens dataset.