Movielens Edx Project

Poonam Quraishy

12/15/2021

Overview

This project is part of the HarvardX Data Science Capstone.

Introduction

The objective of machine learning is to endow computational machines with the intelligence to learn patterns from data which is immensely helpful in analysis and outcome prediction. Recommender systems are a notable class of applications which offer valuable suggestions to users. The objective of this project is to build a movie recommendation system based on user ratings. The dataset used in this project is the 10M version of MovieLens dataset, collected by GroupLens Research.

Aim of the project

The aim of the project is to build a movie recommendation system by training a machine learning algorithm to predict user ratings. The provided 10M MovieLens dataset will be prepared, and split into train and validation sets. A visual and exploratory analysis will be conducted on the data to develop a movie recommendation system.

Modeling approach and definition of RMSE

The usual approach to identify the superior model in machine learning is to define a loss function. The most common approach is to use the mean squared error(MSE) or the root mean squared error(RMSE). The rmse enables us to measure how far predicted values are from observed values.

Importing data from code provided within the course.

The Movielens dataset is downloaded and split into two sets, the edx set which consists of 90% of the data and the validation set consisting of the remaining 10% of the data. The model will be trained on the edx set and evaluated on the validation set.

The edx set will be further divided into a train set consisting of 80% of the data and a test set consisting of 20% of the data. The model will be built and trained using the train and test sets until the desired RSME has been achieved. Finally cross-validation will be conducted using the validation set.

Create edx set, validation set

Note: This script could run for about 10-15 minutes,

run-time may be sooner or later depending on system capability.

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

library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(ggplot2)
library(knitr)
library(kableExtra)
library(tinytex)

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

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

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

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


movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

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

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

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

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

Exploring and analyzing the dataset.

The First entries of the edx training subset.

```
##    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 of the subset.

```
##      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
##  Length:9000055
##  Class :character
##  Mode  :character
##
##
##
```

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

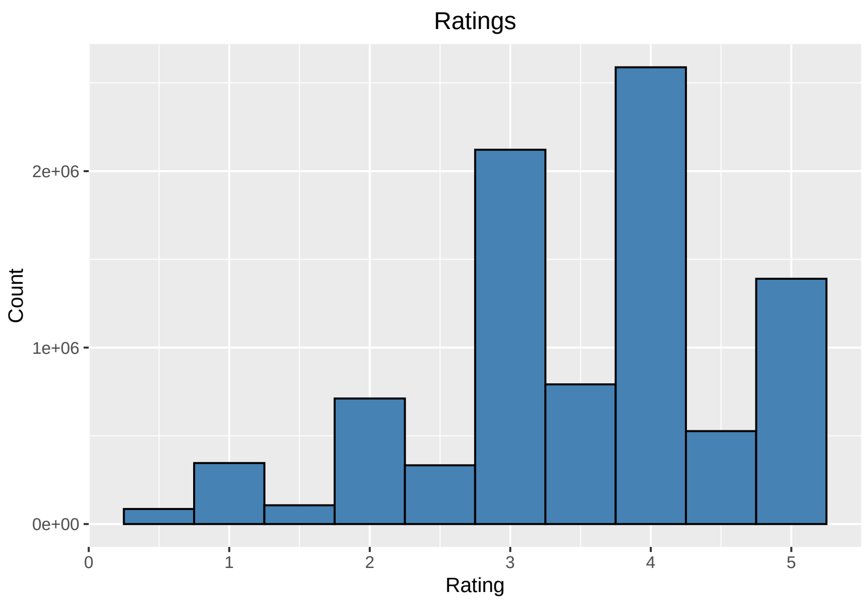
check for NA value

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

Number of unique movies and users in the edx dataset

```
##   un_users un_genres un_movies
## 1    69878       797     10677
```

Plot of the distribution of ratings



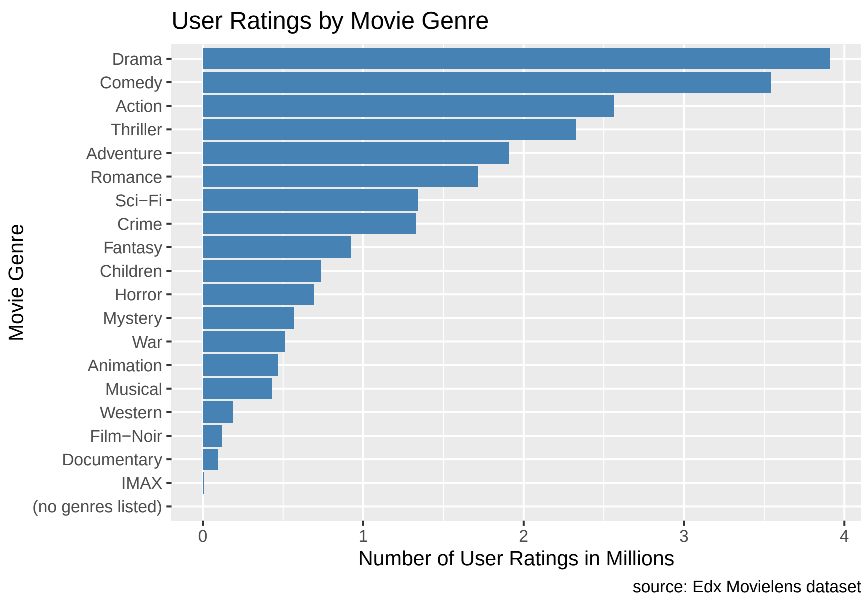
The most rated movies in descending order.

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                         count
##      <dbl> <chr>                                                         <int>
##  1     296 Pulp Fiction (1994)                                           31362
##  2     356 Forrest Gump (1994)                                           31079
##  3     593 Silence of the Lambs, The (1991)                              30382
##  4     480 Jurassic Park (1993)                                          29360
##  5     318 Shawshank Redemption, The (1994)                              28015
##  6     110 Braveheart (1995)                                             26212
##  7     457 Fugitive, The (1993)                                          25998
##  8     589 Terminator 2: Judgment Day (1991)                             25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                              24284
## # ... with 10,667 more rows
```

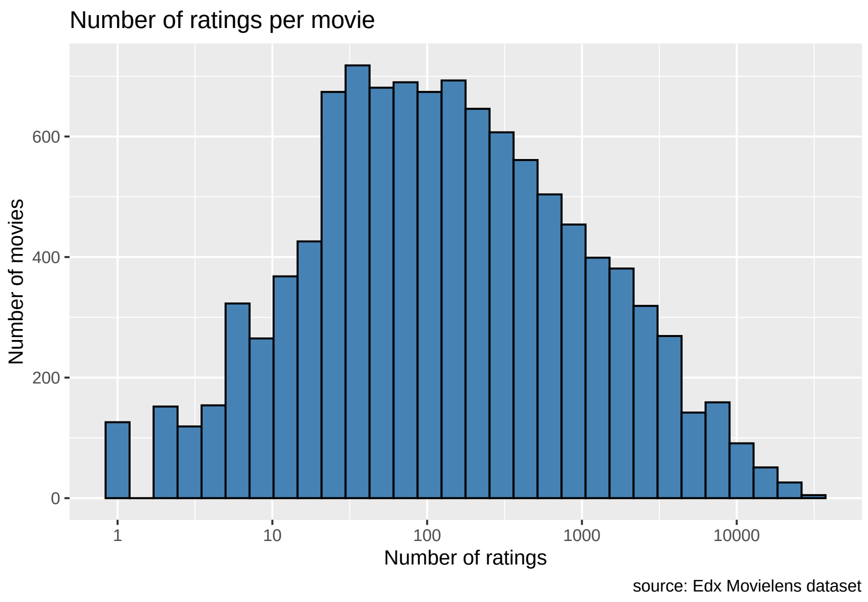
Five most given ratings from most to least.

```
## # A tibble: 5 x 2
##   rating   count
##     <dbl>   <int>
## 1     4    2588430
## 2     3    2121240
## 3     5    1390114
## 4     3.5   791624
## 5     2     711422
```

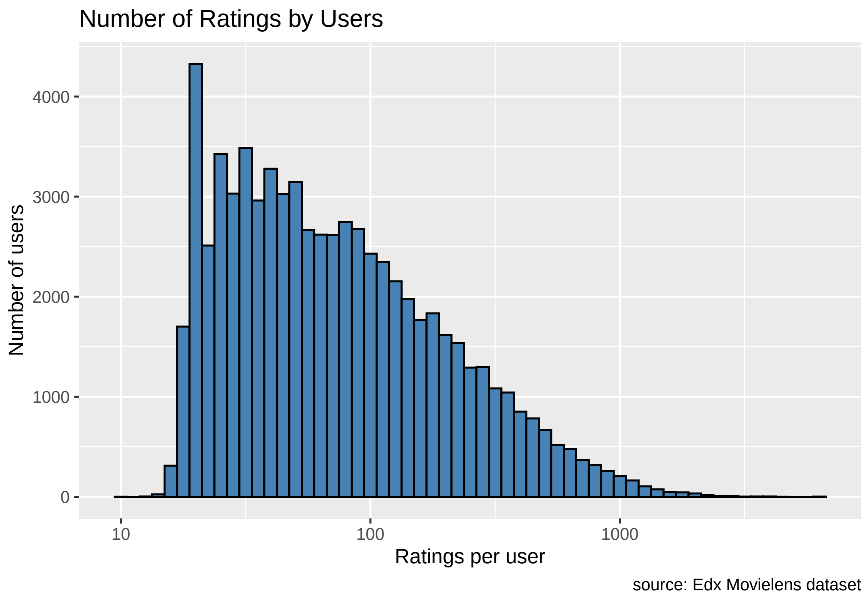
Visualizing user ratings by grouping movies by genre.



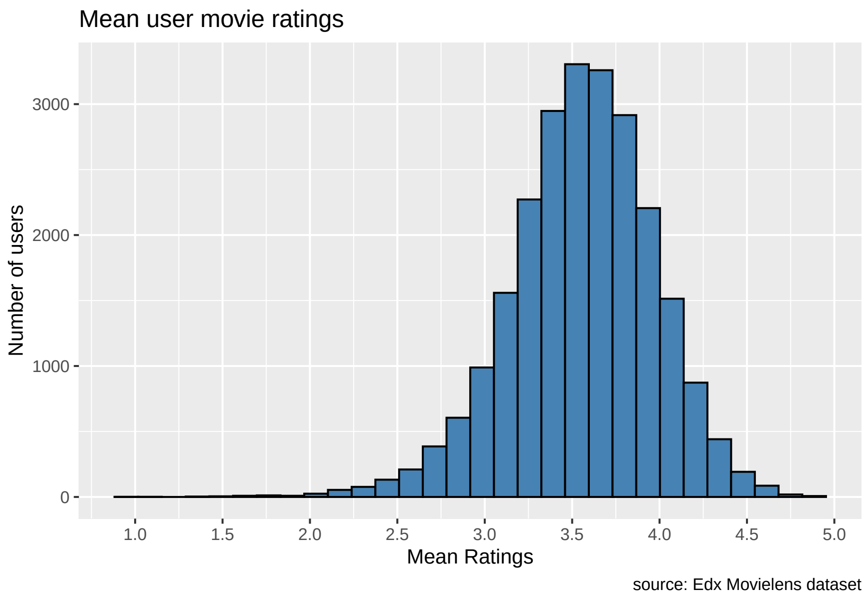
Number of ratings per movie



Number of rating per user



Mean user Ratings.



Data Modeling

The 'Edx' dataset will be randomly split into a train_set and test_test to create multiple models and test their RMSE'S.

Modeling Approach

The models will be tested based on our previously defined RMSE which will enable us to measure how far predicted values are from observed values. The RMSE is our measure of model accuracy. The loss function is defined as follows.

MODEL 1 - Mean Rating Model.

This simplest linear model makes predictions based on the mean rating of the dataset. The model assumes all users will give the same rating, regardless of genre and user without considering any bias. According to statistical theory the average minimizes the RMSE. This simple model is based on the formula

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

. Here μ is the average rating of all the movies, $\epsilon_{u,i}$ is the independent error, and $ Y_{u,i}$ is the prediction.

```
# Taking the average of all observed ratings.
mu <- mean(train_set$rating)
mu
```

```
## [1] 3.512574
```

```
# Results of mean rating model prediction
meanmodel_rmse <- RMSE(test_set$rating, mu)
meanmodel_rmse
```

```
## [1] 1.060704
```

The results of the mean rating model are displayed

```
rmse_results <- data_frame(Method = "Mean Rating Model",Dataset= "Edx_Test", RMSE = meanmodel_rmse)
rmse_results %>% knitr::kable()
```

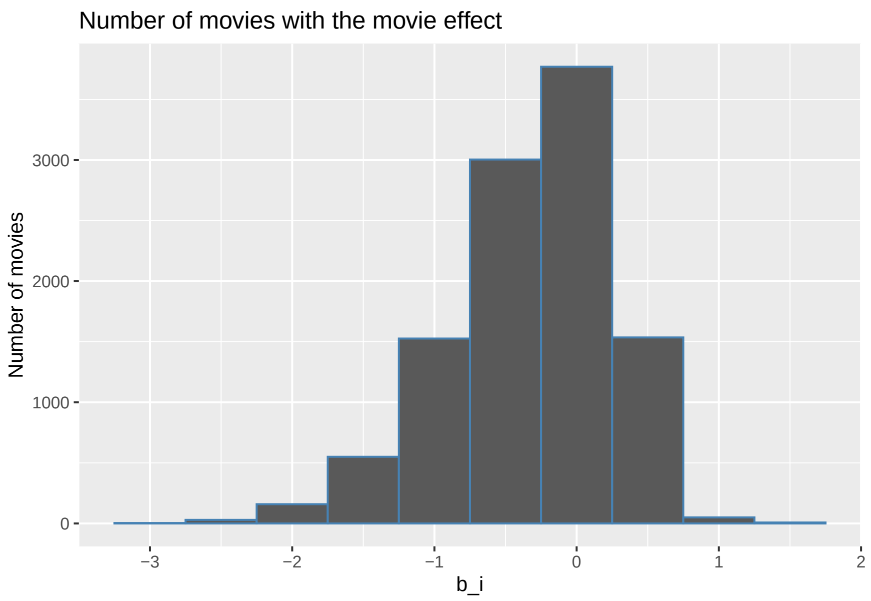
Method	Dataset	RMSE
Mean Rating Model	Edx_Test	1.060704

MODEL 2 - Model with added movie effect.

Movie ratings vary by popularity and generally popular movies receive higher ratings and unpopular movies are rated lower. The simple mean rating model can be improvised by adding a new variable 'b' which is the movie bias for each movie 'i'. The (b_i) can be computed by estimating the deviation of each movie's mean rating from the total mean of all movies μ. We call this the movie effect and the model is based on the formula

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

```r
# Model 2 with added movie effect

movie_averages <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_averages %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("steelblue"),
                         ylab = "Number of movies", main = "Number of movies with the movie effect")
```



```r
b_i <- test_set %>%
  left_join(movie_averages, by='movieId') %>%
  .$b_i
```

This computed bias is the movie effect which is added to the model to improve the prediction.

Test and save rmse results

```r
# Test and save rmse results

predicted_ratings <- mu + b_i
movieeffect_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie effect model",Dataset="Edx_Test",
                                     RMSE = movieeffect_rmse ))

# Check results
rmse_results %>% knitr::kable()
```

Method	Dataset	RMSE
Mean Rating Model	Edx_Test	1.0607045
Movie effect model	Edx_Test	0.9437144

Model 3 - Model with added user and movie effect

There is no predefined standard based on which users rate movies hence there is a lot of variability in the manner in which movies are rated. Some users are highly critical and tend to rate all movies lower while some users generously rate all movies higher. The model can be further improved by adding the user effect (b_u). The formula for the improved model which takes into consideration movie and user effect is as follows

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

```r
user_averages <- train_set %>%
  left_join(movie_averages, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

b_u <- test_set %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  .$b_u

predicted_ratings <- mu + b_i + b_u

# check and save results


usereffect_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie and user effect model",Dataset="Edx_Test",
                                     RMSE = usereffect_rmse))
rmse_results %>% knitr::kable()
```

Method	Dataset	RMSE
Mean Rating Model	Edx_Test	1.0607045
Movie effect model	Edx_Test	0.9437144
Movie and user effect model	Edx_Test	0.8661625

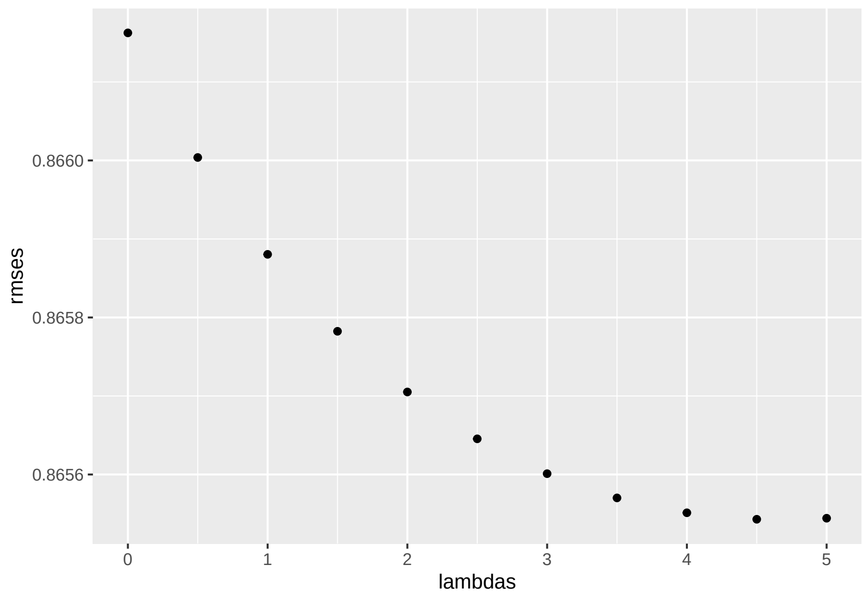
Model 4 - Model With Regularization

Few viewers rate only a small number of movies and tend to give the highest or lowest rating available which strongly influences the error metrics and skews the prediction. Regularization is a method that uses a tuning parameter, λ,which penalizes irregular estimates in order to minimize the RMSE.

```r
lambdas <- seq(0, 5, 0.50)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)

  movie_averages <- train_set %>%
    group_by(movieId) %>%
    summarize(movie_averages = sum(rating - mu)/(n()+l))

  user_averages <- train_set %>%
    left_join(movie_averages, by='movieId') %>%
    group_by(userId) %>%
    summarize(user_averages = sum(rating - mu - movie_averages)/(n()+l))

  predicted_ratings <- test_set %>%
    left_join(movie_averages, by = "movieId") %>%
    left_join(user_averages, by = "userId") %>%
    mutate(pred = mu + movie_averages + user_averages) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})
```

We plot RMSE vs lambdas to select the optimal lambda

```
qplot(lambdas, rmses)
```



```
  lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 4.5
```

For the full model, the optimal lambda is: 5

```
# check and save results

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Model with Regularization",Dataset ="Edx_Test",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

Method	Dataset	RMSE
Mean Rating Model	Edx_Test	1.0607045
Movie effect model	Edx_Test	0.9437144
Movie and user effect model	Edx_Test	0.8661625
Model with Regularization	Edx_Test	0.8655430

Using the Model with Regularization on the hold-out validation dataset.

```r
mu <- mean(validation$rating)
l <- 5
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + l))

b_u <- validation %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() +l))

predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i +  b_u) %>% .$pred

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8403161
```

```r
model_regularization <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method = "Model with Regularization", Dataset="Validation",
                                     RMSE = model_regularization))

rmse_results %>% knitr::kable()
```

Method	Dataset	RMSE
Mean Rating Model	Edx_Test	1.0607045
Movie effect model	Edx_Test	0.9437144
Movie and user effect model	Edx_Test	0.8661625
Model with Regularization	Edx_Test	0.8655430
Model with Regularization	Validation	0.8403161

Conclusion

After testing several models on the train and test set we can infer that adding the movie effect and user effect results in lowering the RMSE. The final chosen model takes into account the movie effect and user effect with regularization to further minimize the RMSE. Testing the model with regularization on the validation set has resulted in an RMSE of 0.84 which is lower than the target RMSE of 0.86. Hence we can conclude that the model with regularization successfully utilizes machine learning techniques to predict ratings.

```r
#### Appendix ####
print("Operating System:")
```

```
## [1] "Operating System:"
```

```r
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.2
## year           2020
## month          06
## day            22
## svn rev        78730
## language       R
## version.string R version 4.0.2 (2020-06-22)
## nickname       Taking Off Again
```