Ph125.9x - Capstone MovieLens

Ken Gustafson

Contents

1 Introduction

This project consists of analyzing the MovieLens dataset, and providing a rating prediction for each movie. The dataset contains six features: userid, movieid, rating, timestamp, movie title, and genre. Everyone who has submitted a rating gets their own unique userId. Each movie has a unique movieid. Each observation has a corresponding rating, which was recorded at the stated timestamp for the given movie title. Each movie is labeled with one or more genres as a multi-valued feature.

The goal of the project is to create a machine learning model which generates predictions of movie ratings. The accuracy of our predictions will be using the root means squared error as the measure of distance to the true rating. The formula is $RMSE = \sqrt{\frac{1}{n}(\sum_{i=1}^{n} y - \hat{y})}$.

There are an adequate number of observations for modeling as there are roughly 10 million rows of data. The key steps that were performed are data exploration and analysis, gathering insights, and model selection.

2 Methods / Analysis

2.1 Constructing the Data and Validation Set

```r
library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(dplyr)
library(tidyr)
library(lubridate)
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")

# if using R 4.0 or later:
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)
val <- validation
```

The above code creates the dataset edx and and a validation set we will use to check how well our final model predicts future movie ratings.

2.2 Exploring the Data

There are a total of 69,878 unique reviewers and 10,677 different movies.

```r
# Computes number of unique users and movies
length(unique(edx$userId))
```

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

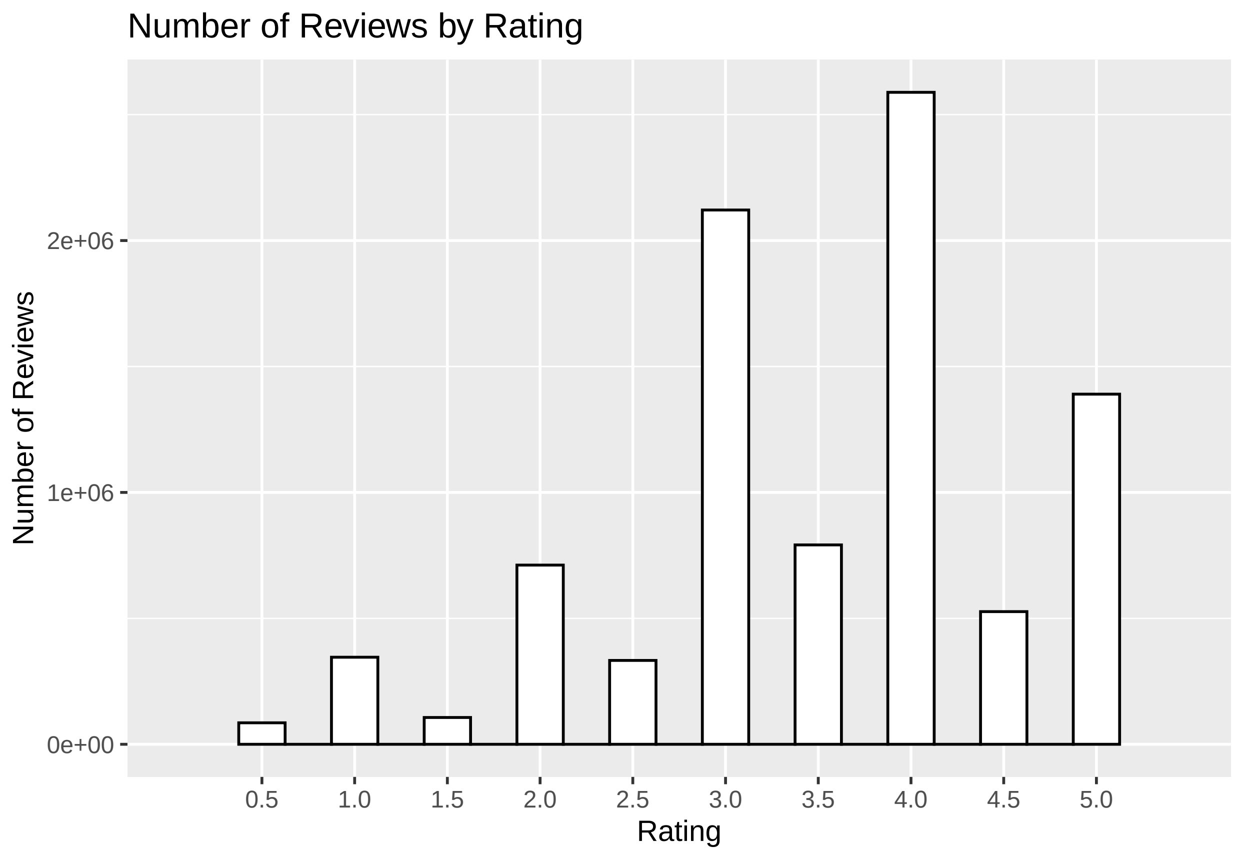
```r
length(unique(edx$movieId))
```

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

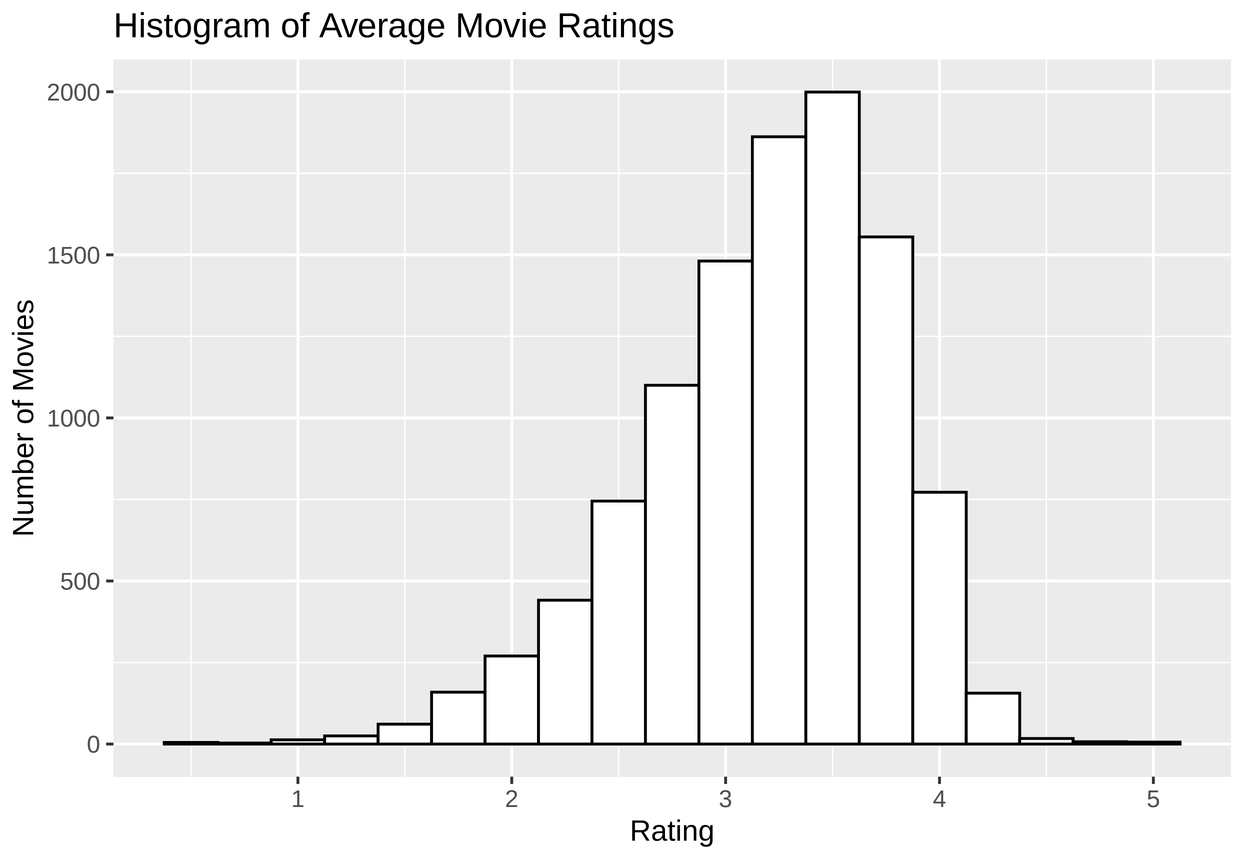
The distribution of the ratings is shown in the following histogram. We can clearly see that reviewers favored giving movies ratings of 3 or 4, and showed a strong preference for giving whole numbers as a rating as opposed to half ratings.

```r
# Histogram for Number of Reviews

edx %>% ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "black",fill="white") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5)))  +
  ggtitle("Rating distribution") +
  labs(title = "Number of Reviews by Rating", x="Rating",y = "Number of Reviews")
```

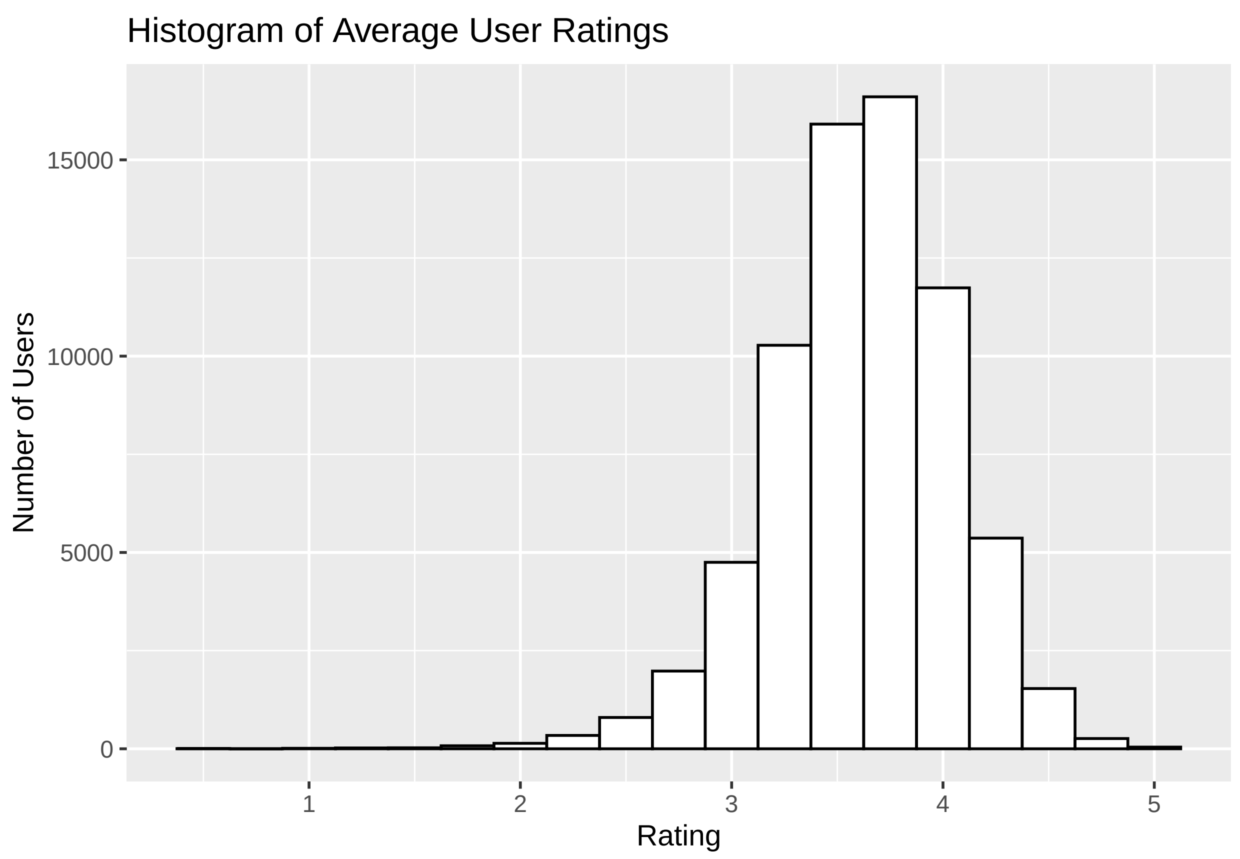


```
# Histogram for Average Movie Rating
meanmovie <- edx %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)
p1 <- ggplot(meanmovie, aes(x=meanm)) + geom_histogram(binwidth = .25,color = "black", fill = "white") +
p1
```



The above histogram shows the distribution of the average movie ratings. The data looks left skewed so the mean of all movies will be less than the median rating. This indicates that most movies are given generally favorable reviews but that there are some moves which are much worse than average and this will drag down the mean. Including the average rating for each movie for our future modeling should increase predictive power as there is much variability between average individual movie ratings.

```
meanuser <- edx %>% group_by(userId) %>% summarize(meanu = mean(rating))
meanuser$userId <- as.integer(meanuser$userId)
p2 <- ggplot(meanuser, aes(x=meanu)) + geom_histogram(binwidth = .25,color = "black", fill = "white") +
p2
```



The above histograms shows the distribution of the average rating each user gave. It looks similar to the average movie histogram but seems more normally distributed around its mean. Including the average rating of each user is another addition we can make to future models as there is a lot of variability of the average ratings between users.

2.3 Model 1

As an initial model, I used the simplest model: an intercept only model where the average rating for all movies is used as the estimate for any new movie rating. In model form it is $Y_i = \mu + \epsilon_i$, $i = 1, \ldots, n$, where ϵ_i is assumed to be normally distributed with mean zero and variance σ^2, i.e. $\epsilon_i \sim N(0, \sigma^2)$.

Below is our function which will compute the RMSE.

```r
# Function to Compute RMSE
rmse <- function(realrating, predictedrating){
  dif <- sqrt(mean((realrating - predictedrating)^2))
  return(dif)
}
```

```r
# Create Test and Validation Set for Model 1
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx1 <- edx[-test_index,]
temp <- edx[test_index,]

validation1 <- temp %>%
      semi_join(edx1, by = "movieId") %>%
      semi_join(edx1, by = "userId")

removed <- anti_join(temp, validation1)
edx1 <- rbind(edx1, removed)

# Model 1
mu <- mean(edx1$rating)

# Model 1 Cross Validation Statistic
model1cvrmse <- rmse(validation1$rating,mu)
model1cvrmse
```

```
## [1] 1.060054
```

```
# Model 1 Final Validation RMSE

predictedrating <- mu

val2 <- validation
val2 <- left_join(val2,meanmovie,by="movieId")
val2 <- left_join(val2,meanuser,by="userId")

predictedrating <- mu
RMSE(predictedrating, validation$rating)
```

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

The estimate for μ is simply the mean of all the movie ratings, which is 3.512. Computing the cross validation statistic yields CV = 1.061. All future analysis should have a CV lower than this value as this is the best we can achieve so far.

2.4 Model 2

To further refine model 1, we can start incorporating variables from the edx dataset. Since not all movies are created equally, the average rating for each individual movie will vary. To add the information for how the ratings differ between each movie, the new model is $Y_{i,j} = \mu + m_j + \epsilon_{i,j}$,$m = 1, \ldots, d$ where d is the total number of distinct movies. m_j is the amount each individual movie differs from the overall mean. The CV statistic for this model is .943.

```
# Model 2
meanmovie <- edx1 %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)

meanmovie$mi <- meanmovie$meanm - mu

val2 <- validation1
val2 <- left_join(val2, meanmovie, by="movieId")


# Model 2 Cross Validation Statistic
model2rmse <- rmse(validation1$rating, val2$meanm)
model2rmse
```

```
## [1] 0.9429615
```

```
# Model 2 Final Validation RMSE

predictedrating <- mu + val2$mi

meanmovie <- edx %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)

meanmovie$mi <- meanmovie$meanm - mu


val2 <- validation
val2 <- left_join(val2,meanmovie,by="movieId")
val2 <- left_join(val2,meanuser,by="userId")

predictedrating <- mu + val2$mi
RMSE(predictedrating, validation$rating)
```

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

2.5 Model 3

Another approach is to use a model that can incorporate people's different habits when giving out ratings. Some people can be easily pleased and will tend to give out higher ratings for all movies while others are more pessimistic and give out lower ones. This model equation is $Y_{i,k} = \mu + +u_k + \epsilon_{i,k}$ $k = 1, \ldots, f$ where f is the number of unique users. The calculated CV is .978.

```
# Create Test and Validation Set for Model 3

#test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
#edx1 <- edx[-test_index,]
#temp <- edx[test_index,]

#validation1 <- temp %>%
 #     semi_join(edx1, by = "movieId") %>%
  #    semi_join(edx1, by = "userId")

#removed <- anti_join(temp, validation1)
#edx1 <- rbind(edx1, removed)


# Model 3

meanuser <- edx1 %>% group_by(userId) %>% summarize(meanu = mean(rating))
meanuser$userId <- as.integer(meanuser$userId)

meanuser$ui <- meanuser$meanu - mu

val3 <- validation1
val3 <- left_join(val3, meanuser, by="userId")
```

```
# Model 3 Cross Validation Statistic
model2rmse <- rmse(validation1$rating, val3$meanu)
model2rmse
```

```
## [1] 0.977709
```

```
# Model 3 Final Validation RMSE

predictedrating <- mu + val3$ui

meanmovie <- edx %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)

meanuser <- edx %>% left_join(meanmovie, by="movieId") %>% group_by(userId) %>% summarize(ui=mean(rating
meanuser$userId <- as.integer(meanuser$userId)

val2 <- validation
val2 <- left_join(val2,meanmovie,by="movieId")
val2 <- left_join(val2,meanuser,by="userId")

predictedrating <- mu + val2$ui
RMSE(predictedrating, validation$rating)
```

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

2.6 Model 4

We can combine the previous two ideas to create a model that combines both the effect of people and of the movie on ratings. This new model is now $Y_{i,j,k} = \mu + m_j + u_k + \epsilon_{i,j,k}$. The calculated CV is .865. As expected, since this combines the benefits of both model 2 and model 3, it has a lower CV statistic than either individual model.

```
# Model 4

meanmovie <- edx1 %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)

meanmovie$mi <- meanmovie$meanm - mu

meanuser <- edx1 %>% left_join(meanmovie, by="movieId") %>% group_by(userId) %>% summarize(ui=mean(rati
meanuser$userId <- as.integer(meanuser$userId)

val4 <- validation1
val4 <- left_join(val4,meanmovie,by="movieId")
val4 <- left_join(val4,meanuser,by="userId")


# Model 4 Cross Validation Statistic
predictedrating <- mu + val4$mi + val4$ui
RMSE(predictedrating, validation1$rating)
```

```
## [1] 0.8646843
```

```
# Model 4 Final Validation RMSE

predictedrating <- mu + val4$mi + val4$ui

meanmovie <- edx %>% group_by(movieId) %>% summarize(meanm = mean(rating))
meanmovie$movieId <- as.integer(meanmovie$movieId)

meanmovie$mi <- meanmovie$meanm - mu

meanuser <- edx %>% left_join(meanmovie, by="movieId") %>% group_by(userId) %>% summarize(ui=mean(ratin
meanuser$userId <- as.integer(meanuser$userId)

val2 <- validation
val2 <- left_join(val2,meanmovie,by="movieId")
val2 <- left_join(val2,meanuser,by="userId")

predictedrating <- mu + val2$mi + val2$ui
RMSE(predictedrating, validation$rating)
```

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

2.7 Model 5

For this model we will combine regularization with model 4. First we need to find the optimal lambda value which is always a priority when doing nonparametric regression. This optimal value will minimize the RMSE by making the model learn less from outliers and more from data that follows the general pattern. After checking lambda values between 0 and 10 incrementing by 1, we find that a value of 5 minimizes the RMSE for the test data set. Using this optimal lambda value gives us a CV statistic of .86482, our lowest yet. Thus we can conclude that adding regularization improves model 4, albeit very slightly.

```
testlambdas <- seq(0,10,1)
testrmse <- sapply(testlambdas, function(x){
  mu <- mean(edx1$rating)

  meanmovie <- edx1 %>%
    group_by(movieId) %>%
    summarize(meanmovie = sum(rating - mu)/(n() + x))

  meanuser <- edx1 %>%
    left_join(meanmovie, by='movieId') %>%
    group_by(userId) %>%
    summarize(meanuser = sum(rating - meanmovie - mu)/(n() +x))

  predicted_ratings <- validation1 %>%
    left_join(meanmovie, by = "movieId") %>%
    left_join(meanuser, by = "userId") %>%
    mutate(pred = mu + meanmovie +  meanuser) %>% .$pred

return(RMSE(predicted_ratings, validation1$rating))
})
```

```
optimal_lambda <- testlambdas[which.min(testrmse)]
optimal_lambda
```

```
## [1] 5
```

```
min(testrmse)
```

```
## [1] 0.8641362
```

```
# Model 5 Cross Validation
functionrmse <- function(x){
  mu <- mean(edx1$rating)

  meanmovie <- edx1 %>%
    group_by(movieId) %>%
    summarize(meanmovie = sum(rating - mu)/(n() + x))

  meanuser <- edx1 %>%
    left_join(meanmovie, by='movieId') %>%
    group_by(userId) %>%
    summarize(meanuser = sum(rating - meanmovie - mu)/(n() +x))

  predicted_ratings <- validation1 %>%
    left_join(meanmovie, by = "movieId") %>%
    left_join(meanuser, by = "userId") %>%
    mutate(pred = mu + meanmovie +  meanuser) %>% .$pred

return(RMSE(predicted_ratings, validation1$rating))
}

functionrmse(5)
```

```
## [1] 0.8641362
```

```
# Model 5 Final Validation RMSE

functionrmse <- function(x){
  mu <- mean(edx$rating)

  meanmovie <- edx %>%
    group_by(movieId) %>%
    summarize(meanmovie = sum(rating - mu)/(n() + x))

  meanuser <- edx %>%
    left_join(meanmovie, by='movieId') %>%
    group_by(userId) %>%
    summarize(meanuser = sum(rating - meanmovie - mu)/(n() +x))

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

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

functionrmse(5)
```

```
## [1] 0.8648177
```

3 Results

Model	Cross Validation	RMSE Using Validation Set
Intercept Model	1.060	1.0612
Average Movie Model	.9430	.9439
Average User Model	.9777	.9783
Average Movie and User Model	.8647	.8653
Average Movie and User Model with Regularization	.8641	.8648

Based on the cross-validation that was calculated for each model, the final model will be Model 5 which incorporates both the movie and user effect along with regularization. Using the Validation set on our final model to estimate the RMSE, we find a value of .86482. I also ran the validation set on our other models to see what their RMSE would be, but there was very little difference to the estimate calculated using cross validation. This model's predictive power is stronger than the other four as the RMSE value is lower.

4 Conclusion

This analysis was designed to create a model which can predict future ratings about movies. The final and best model was determined to be model five, as it had the best cross validation statistic of all models. Using the validation set the RMSE was calculated to be .86482. This means that on average the squared difference between the predicted rating for the movie and the actual rating for the movie is .86482. A major limitation of this model is that it is only applicable to movies inside the edx dataset as well as the users who submitted the ratings. In order to predict movie ratings for new movies and new users, one approach would be to use linear regression where we could use information about the movie such as the genre or run time to see how that would affect the rating. Future work could also incorporate user data such as gender of the reviewer or their age. Doing linear regression in this manner would let us know what variables are significant in affecting the rating of movies and we would not be limited to only movies and users in the training set.