HarvardX: PH125.9x Data Science MovieLens Capstone Project

Sarma Nimishakavi

1/31/2021

Introduction

MovieLens Capstone Project is implemented for HarvardX: PH125.9x Data Science: Capstone course. As a part of this project we will create a movie recommendation system by applying some of data science concepts learnt during this course. We will:

- Explore MovieLens data provided
- Develop a machine learning algorithm
- Predict ratings using various models
- Evaluate our predictions and adjust models
- Summarize results

Objective

The objective is to develop a movie recommendation system using machine learning algorithm that predicts user ratings and evaluate our model's performance using Root Mean Square Error, or RMSE. RMSE is a frequently used measure of the differences between values (sample or population values) predicted by a model and is a measure of accuracy to compare forecasting errors of different models. RMSE is always non-negative, and a value of 0 would indicate a perfect fit to the data. In general, a lower RMSE is better than a higher one. Formula for computing RMSE for ratings and their corresponding predictors is:

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

```
RMSE <- function(actual , prediction){
  sqrt(mean((actual - prediction)^2))
}
```

Data

For this project, we will be using the MovieLens dataset provided for this course. This is just a small subset of a much larger dataset with millions of ratings. We use the MovieLens 10M dataset that consists of 10 million ratings applied to 10,000 movies by 72,000 users. Code to load this data was already provided for this course.

Data Preparation

In the given dataset many movies are categorized under multiple genres. We will make folowing modifications to the original code. First, we will separate multi-genre rows as separate rows by each individual genre using "separate_rows" function. (*note: I also tested data without separting genres in to multiple rows by commenting out seprate_rows method code. I kept separate_rows code as it gave better results*). Second,

we extract the release year of the movie from the movie title in the data. The following lines of code show modifications to the original code provided by staff.

movielens <- movielens %>% separate_rows(genres, sep = "\|") %>% mutate(releaseyear = as.numeric(str_sub(title,-5,-2)))

```
##Load Data
# 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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

movielens <- movielens %>%
# We will analyse by splitting multi-genres into separate rows
separate_rows(genres, sep = "\\|")%>%
# We will also extract movies release year from Title
mutate(releaseyear = as.numeric(str_sub(title,-5,-2)))
```

We further split movielens into two data sets. "edx", or the training set, will have 90% of data from MovieLens data. This set will be used for training our algorithm. "Validation" or the test set will have 10% of data from Movielens data. We will use this test set for testing movie ratings.

```
# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- as.vector(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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres", "releaseyear")
```

```
edx <- rbind(edx, removed)

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

Exploratory Data Analysis

We start our data analysis by looking at the edx dataset. We can see that variables in this dataset are "userID", "movieID", "rating", "timestamp", "title", and "genres", and "release year" (added by us). Each row represents a rating given to a movie by a user. We also look at distinct number of Movies and Users in the set. We will also look at some statistics like which movies were rated more times and which movies were rated less times. Higher number of ratings may average out over all rating for a movie or fewer ratings may result in too high or too low over all rating in some cases.

```
##   userId movieId rating timestamp                title   genres releaseyear
## 1      1     122      5 838985046     Boomerang (1992)  Romance        1992
## 2      1     185      5 838983525      Net, The (1995)   Action        1995
## 3      1     185      5 838983525      Net, The (1995)    Crime        1995
## 4      1     185      5 838983525      Net, The (1995) Thriller        1995
## 5      1     231      5 838983392 Dumb & Dumber (1994)   Comedy        1994
## 6      1     292      5 838983421      Outbreak (1995)   Action        1995
```

```
# Movies which were rated more
ratedmore <- edx %>% group_by(title) %>%
  summarize(numratings = n()) %>%
  arrange(desc(numratings))
head(ratedmore)
```

```
## # A tibble: 6 x 2
##   title                numratings
##   <chr>                     <int>
## 1 Forrest Gump (1994)      123676
## 2 Toy Story (1995)         119021
## 3 Jurassic Park (1993)     117568
## 4 True Lies (1994)         114120
## 5 Aladdin (1992)           105978
## 6 Batman (1989)             97318
```

```
# MOvies that were rated once
edx %>% group_by(title)%>%
  summarize(numratings = n()) %>%
  filter(numratings==1) %>%
  count() %>% pull()
```

```
## [1] 72
```

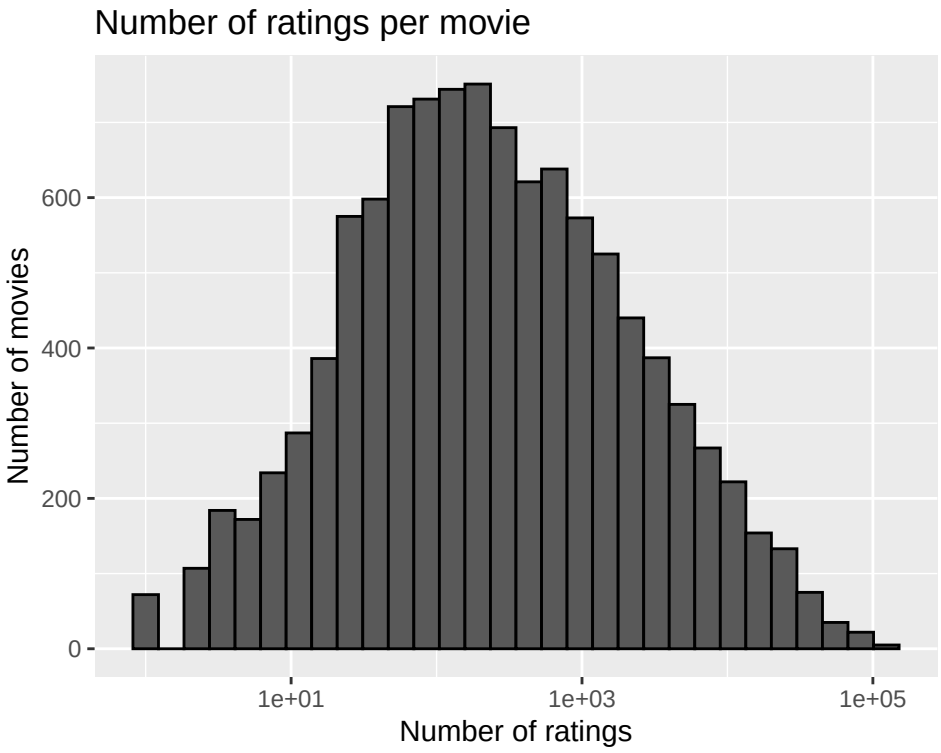
```
# MOvies that were never rated
edx %>% group_by(title)%>%
  summarize(numratings = n()) %>%
  filter(numratings<1) %>%
  count() %>% pull()
```

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

Movie Effect

When we plot the number of ratings per movie, we can observe that some movies are rated more often than others. For example, movies that are advertised better or movies that have popular actors may have higher viewership. This may indicate potential movie bias.

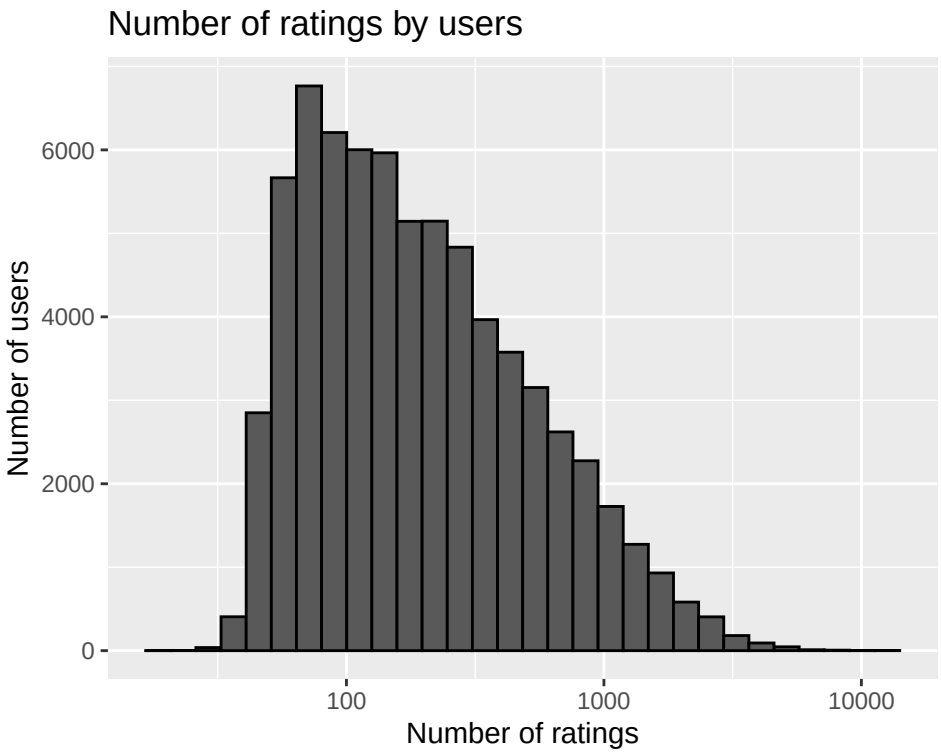
```
edx %>%
count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
  ylab("Number of movies") +
ggtitle("Number of ratings per movie")
```



User Effect

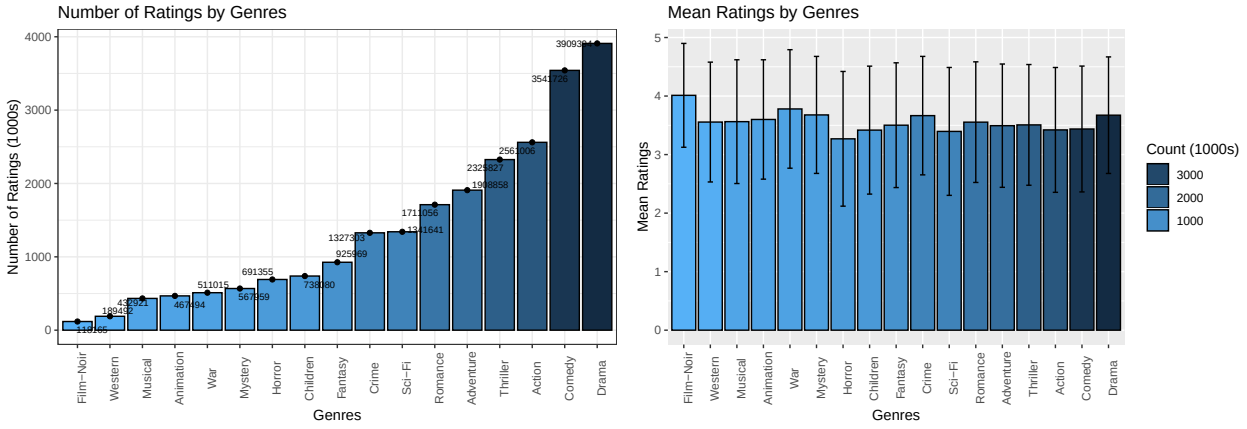
Similarly, we can observe that users may have a preference for some types of movies and may tend to give ratings based on their likings. We will have to evaluate potential user bias as well.

```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings by users")
```



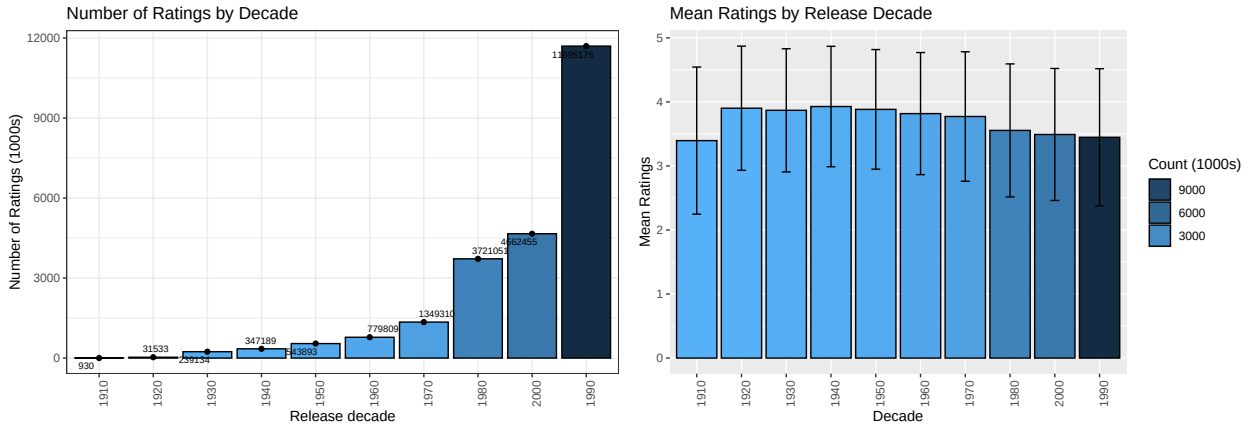
Genre Effect

We want to check if genre of a movie has any affect on rating. We will group validation data by genres an calculate mean and standard deviation for each group. We will also filter all genres that have fewer than 100,000 ratings. We will plot the number of ratings per genre first, and then plot error bars for mean and standard deviation for each group. As we can see from below, Drama and Comedy seem to receive more ratings and Film Noir and War seem to recieve fewer ratings. However, mean ratings for Film Noir are the highest and War is very close to Drama. when we use genres as given in the original set, we can see similar results. Drama gets more ratings and Action|Crime|thriller get fewer ratings, but mean ratings for Action|Crime|thriller are comparable to Drama. Data seems to indicate that genre does affect ratings in general. We will verify this by calculating RMSE.



Release Year Effect

Similarly, we want to check if release year of a movie has any affect on rating. For convenience, we will group validation data by release decades and calculate the mean and standard deviation for each group. We will plot the number of ratings per release decade first. Then, we will plot error bars for mean and standard deviation for each group. As we can see from below, those movies released earlier in the century to middle of the century had fewer ratings. However, the mean ratings for those movies are similar or a bit better from the 50s to the 80s. Data seems to indicate that the release year does affect ratings in general. We will verify this by calculating RMSE.



Methodolody and Approach

We recall from our data analysis that the MovieLens dataset originally had 6 variables, viz. "userID", "movieID", "rating", "timestamp", "title", and "genres". We added seventh variable "releaseyear" by capturing this information from "title". Rating is what we are trying to predict; therefore we need to find out what other variables can explain variation in ratings. Some variables may or may not have any effect in rating. We will start with an assumption that other variables do not have any effect on rating and use the mean of all ratings to predict ratings. After this baseline prediction, we will add biases induced by other variables to our baseline model, one at a time. We will find RMSE for our model with each bias added. Finally we will regularize our algorithm and see if it improves our RMSE. We will recommend the model with the lowest RMSE to be used for predictions.

Average Model

The baseline model uses the average rating of all ratings in the dataset and assumes that ratings are just random variation and other variables like Users do not have any affect.

$$Y_{u,i} = \mu, \tag{1}$$

where $Y_{u,i}$ is the predicted rating of user u and movie i and μ is the average rating across all the ratings. Mean in this case is 3.512 (`mean(edx$rating)`). We get RMSE for this model as 1.0513757. When we comment out split_rows code and run with genres as given in the original set we get RMSE of 1.0612018 for average model.

```
mu <- mean(edx$rating)
naive_rmse<- RMSE(validation$rating, mu)

rmse_tbl <- tibble(method = "Average Model", RMSE = naive_rmse)
rmse_tbl %>% knitr::kable(booktabs = T) %>%
            kableExtra::kable_styling(latex_options = "basic", full_width = F)
```

method	RMSE
Average Model	1.051376

Movie Effect Model

During our exploratory data analysis we noticed movie bias. In other words, some movies are liked better in general and tend to receive higher ratings. We can add an independent error term b_i for movie bias. This term averages the rankings for any movie i. Our new model is:

$$Y_{u,i} = \mu + b_i. \tag{2}$$

```
# add movie bias term, b_i
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
# predict all unknown ratings with mu and b_i
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
# calculate RMSE of movie ranking effect
movie_rmse <-RMSE(validation$rating, predicted_ratings)

rmse_tbl <- bind_rows(rmse_tbl,tibble(method = "Movie Effect Model", RMSE = movie_rmse))
rmse_tbl %>% knitr::kable(booktabs = T) %>%
             kableExtra::kable_styling(latex_options = "basic", full_width = F)
```

method	RMSE
Average Model	1.051376
Movie Effect Model	0.940319

Movie and User Effect Model

We can now introduce the user bias term b_u in order to include user bias. This term minimizes the effect of extreme ratings made by users based on their likings or dislikings. Each user u is given a bias term. Our new model is:

$$Y_{u,i} = \mu + b_i + b_u. \tag{3}$$

```
# add user bias term, b_u
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
# predict new ratings with movie and user bias
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 movie and user bias effect
movie_user_rmse <-RMSE(predicted_ratings, validation$rating)
rmse_tbl <- bind_rows(rmse_tbl,tibble(method = "Movie and User Effect Model",
                                      RMSE = movie_user_rmse))
rmse_tbl %>% knitr::kable(booktabs = T) %>%
             kableExtra::kable_styling(latex_options = "basic", full_width = F)
```

method	RMSE
Average Model	1.051376
Movie Effect Model	0.940319
Movie and User Effect Model	0.856977

Movie, User and Genres Effect Model

We can now introduce the genres bias term b_g in order to include genre bias. This term minimizes the effect of extreme ratings and more frequent ratings for popular genres. Each genre g is given a bias term. Our new model is:

$$Y_{u,i} = \mu + b_i + b_u + b_g. \tag{4}$$

```
# compute genre bias term, b_g
b_g <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i-b_u))

# predict new ratings with movie,user and genre bias
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_g, by='genres') %>%
  mutate(pred = mu + b_i + b_u+b_g) %>%
  pull(pred)

# calculate RMSE of movie, user and genre ranking effect
movie_user_genre_rmse <-RMSE(predicted_ratings, validation$rating)
rmse_tbl <- bind_rows(rmse_tbl,tibble(method = "Movie, User and Genre Effect Model",
                                      RMSE = movie_user_genre_rmse))
rmse_tbl %>% knitr::kable(booktabs = T) %>%
             kableExtra::kable_styling(latex_options = "basic", full_width = F)
```

method	RMSE
Average Model	1.0513757
Movie Effect Model	0.9403190
Movie and User Effect Model	0.8569770
Movie, User and Genre Effect Model	0.8568871

note: when we use genres without split_rows, for this model we get RMSE of 0.8649469

Movie, User, Genres and Year Effect Model

We can now introduce the release year bias term b_y in order to include release year bias. This term minimizes the effect of extreme ratings and more frequent ratings for movies by release years. Each release year y is given a bias term. Our new model is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_y. \tag{5}$$

```
# compute year bias term, b_y
b_y <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_g, by='genres') %>%
  group_by(releaseyear) %>%
  summarize(b_y = mean(rating - mu - b_i-b_u-b_g))

# predict new ratings with movie,user, genre and year bias
predicted_ratings <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_g, by='genres') %>%
  left_join(b_y, by='releaseyear') %>%
  mutate(pred = mu + b_i + b_u+b_g+b_y) %>% #+
  pull(pred)

# calculate RMSE of user, movie, genre, and year ranking effect
movie_user_genre_year_rmse <-RMSE(predicted_ratings, validation$rating)
rmse_tbl <- bind_rows(rmse_tbl,tibble(method = "Movie, User, Genre and Year Effect Model",
                                      RMSE = movie_user_genre_year_rmse))
rmse_tbl %>% knitr::kable(booktabs = T) %>%
             kableExtra::kable_styling(latex_options = "basic", full_width = F)
```

method	RMSE
Average Model	1.0513757
Movie Effect Model	0.9403190
Movie and User Effect Model	0.8569770
Movie, User and Genre Effect Model	0.8568871
Movie, User, Genre and Year Effect Model	0.8565383

note: when we use genres without split_rows, for this model we get RMSE of 0.8647606

Regularized Movie, User, Genres and Year Effect Model

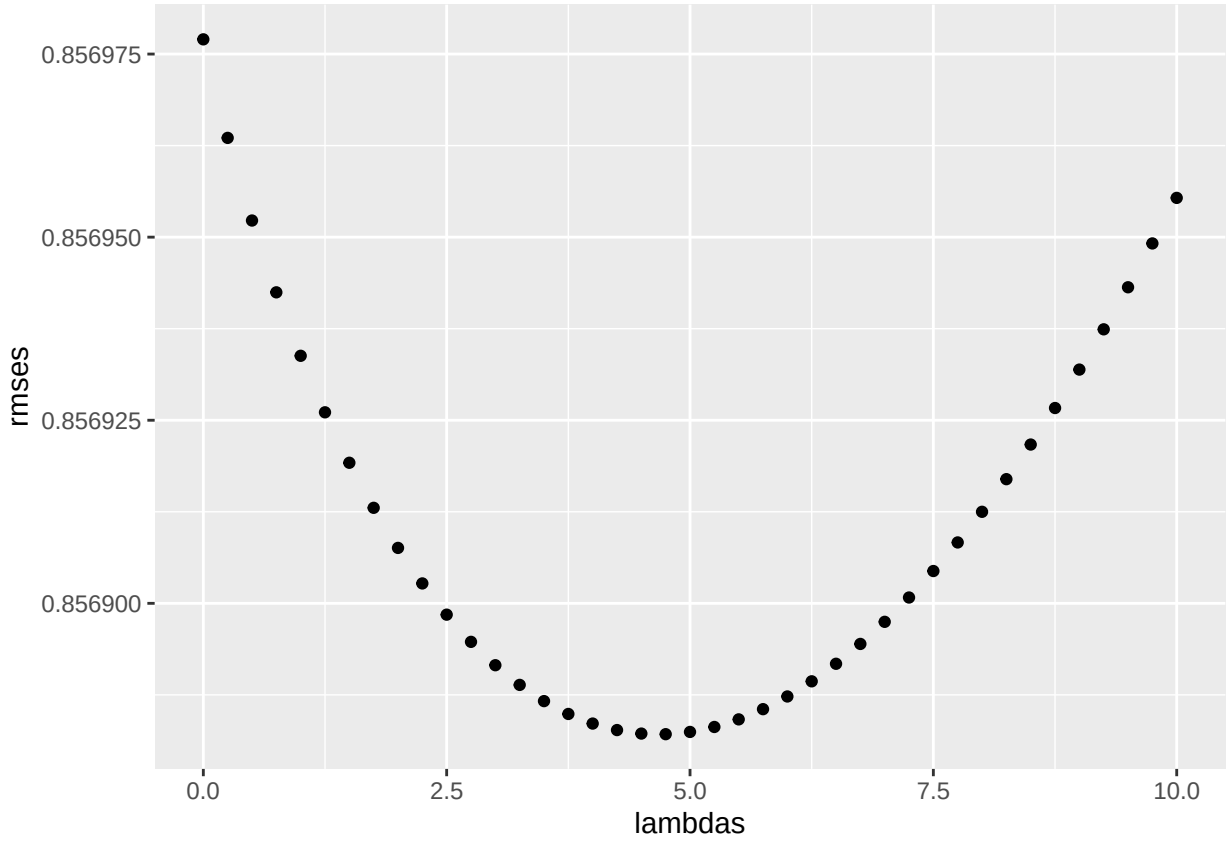
Some models may overfit the traning set but have high variance in predictions. We want to minimize the variance in prediction by avoiding overfitting. Regularization is a commonly used technique for this. Regularizaion penalizes incorrect estimates. We will apply regularization on both b_i,b_u,b_g and b_y.

After adding regularization our new model is:

$$\frac{1}{N}\sum_{u,i}(Y_{u,i} - \mu - b_i - b_u - b_g)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2 + \sum_g b_y^2), \tag{6}$$

where the first term is same as least squares equation and the last term is the penalty with large bias terms. We will try to minimize biases using a single λ. We will test `lamda <- seq(from=0, to=10, by=0.25)` and plot the results below:

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



We see that the minimizing λ term is

```
lambdas[which.min(rmses)]
```

```
## [1] 4.75
```

Following code executes our final model.

```
# choose minimized lambda
lam <- lambdas[which.min(rmses)]
# compute regularize movie bias term
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lam))
# compute regularize user bias term
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lam))
b_g <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  group_by(genres) %>%
```

```
  summarize(b_g = mean(rating - b_i-b_u-mu)/(n()+lam))
b_y <- edx %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  left_join(b_g, by='genres') %>%
  group_by(releaseyear) %>%
  summarize(b_y = mean(rating - b_i-b_u-b_g-mu)/(n()+lam))
# compute predictions on validation set based on these above terms
predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "releaseyear") %>%
  mutate(pred = mu + b_i + b_u+b_g+b_y) %>%
  pull(pred)
# output RMSE of these predictions
regularized_movie_user_genre_year_rmse <-RMSE(predicted_ratings, validation$rating)
rmse_tbl <- bind_rows(rmse_tbl, tibble(method = "Regularized Movie, User, Genre and Year Effect Model",
                                       RMSE = regularized_movie_user_genre_year_rmse))
```

Results

The RMSE values of all the our models are tabluated as follows:

```
lowestRME <-which.min(rmse_tbl$RMSE)

rmse_tbl %>% knitr::kable(booktabs = T) %>%
  kableExtra::kable_styling(latex_options = "basic", full_width = F) %>%
  kableExtra::row_spec(lowestRME, bold = T, color = "black", background = "yellow")
```

method	RMSE
Average Model	1.0513757
Movie Effect Model	0.9403190
Movie and User Effect Model	0.8569770
Movie, User and Genre Effect Model	0.8568871
Movie, User, Genre and Year Effect Model	**0.8565383**
Regularized Movie, User, Genre and Year Effect Model	0.8568821

I came across a couple of interesting observations. The best RMSE was for non-regularized model. This is for model "Movie, User, Genre and Year Effect Model" and the value is 0.8565383. When I applied Regularization for this model, I got a RMSE of 0.8568821. Out of curiosity, I applied Regularization for the model without year ("Movie, User, Genre Effect Model"). This model also got RMSE of 0.8568821. I did not include the code for "Regularized Movie, User, Genre Effect Model" as it is giving me same result (0.8568821). This needs to be further researched. Probably with 4 bias variables, our variance and bias reached a balance that Regularization is not improving. When I ran these models with out splitting genres, I got RMSE of 0.8647606 for "Movie, User, Genre and Year Effect Model" and 0.8648164 as RMSE after applying regularization on this model. This in inline with above observation that RMSE did not improve with regularization with final model. In both approaches (splitting genres (0.8565383) and not splitting (0.8647606)), I got better RMSE than goal RMSE of 0.86490.

Conclusion and Future Work

We built a machine learning algorithm to predict movie ratings with the MovieLens dataset using regression models, including movie, user, genre and release year bias and applying regularization. We may have to investigate further as to why regularization on our 4 variable bias actualy slightly worsened. We may go beyond simple regression models and try random forest method, PCA or Matrix factorization to further improve our predictions. These methods may require more memory and machine power. I will try to apply some of these techniques for second capstone project as part of this curriculum.

References

1. Rafael A.Irizarry (2019), [Introduction to Data Science: Data Analysis and Prediction Algorithms with R] (https://rafalab.github.io/dsbook)
2. Yixuan Qiu (2017), [recosystem: recommendation System Using Parallel Matrix Factorization] (https://cran.r-project.org/web/packages/recosystem/vignettes/introduction.html)
3. Koren, Y. (2009), [The bellkor solution to the netflix grand prize. Netflix prize documentation, 81, 1-10]. (https://netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf)