Movielens-report-Jersson Placido

Jersson Emir Placido Escobar

2021-04-02

1. INTRODUCTION

Recommendation systems utilize ratings that users have given to certain movies or items to predict what rating a given user will give to that item or film. The software from the company offering the movie or the item will recommend the item or movie to the user based on how high the predicted rating is.

As data from big entertainment and consumption corporations such as netflix or amazon are not available, the GroupLens research lab generated their own database (MovieLens) with more than 20 million ratings for more than 27.000 movies and 130.000 users. as recommendation systems are required by the entertainment industry, it is necessary to develop recommendation systems able to detect and recommend movies to the users in a faster and precise way.

This report include the development of a movie recommendation system using the MovieLens dataset. To reduce computation time a 10M version of the MovieLens database was used. The recommendation system was developed and tested using root mean squared error (RMSE). The recommendation systems was developed using the mean of the ratings for the different predictors. The predictors evaluated were, movieID, UserID, date and genres. The influence of the predictors was tested using the RMSE and the model with the lowest RMSE was selected. Regularization was applied to the model to reduce the bias in the model.

The report include a step by step description of the recommendation system since the initial data partition until the final validation step.

2. METHODS/ANALYSIS

2.1 Create the train set (edx) and test set (validation set| final hold-out test set)

The first step to create the validation systems is the creation of the train set and test set from the 10M MovieLens data set. This data set can be found in the following link: http://files.grouplens.org/datasets/movielens/ml-10m.zip

the libraries required for developing the train and test set from the movielens 10M dataset are described next:

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
```

The process to download the files and organize the content of the initial database **movielens** is described next:

```
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")
```

To create the recommendation system was necessary to create the train and test sets from the movielens database previously defined. The train test was identified as **edx**, meanwhile, the test set was defined as temp during this step. the test set is defined as 10% of the initial **movielens** database.

```
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,]
```

As it is important for the model structure and development that both the userId and movieId included in the test set are includes in the **edx** set. the following code create the **validation** set which is the test set including the userId and movieid in the edx set.

```
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

As some rows were removed from the validation set, this rows were added back to the edx set. using the next code.

```
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

2.2 Creation of the test and train sets from the original train (edx) dataset

The train (edx_train_set) and test (edx_test_set) sets from the **edx** that were utilized to experiment and evaluate different parameters were produced with the following set. in this case the train set is defined as 80% of the original edx data

```
set.seed(1, sample.kind="Rounding")
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
edx_train_set<- edx[-edx_test_index,]
edx_test_set <- edx[edx_test_index,]
```

Similar to the original dataset, the train and test sets generated from edx were modified to include the movieid and userid in the test set in the train set. Additionally, the rows removed from the test set were included in the train set. This process was carried out with the following text

```
validationedx <- edx_test_set %>%
  semi_join(edx_train_set, by = "movieId") %>%
  semi_join(edx_train_set, by = "userId")

removed_edx <- anti_join(edx_test_set, validationedx)
edx_train_set<- rbind(edx_train_set, removed_edx)
```

The RMSE equation followed the equation provided in the section 33.7.3 of the book

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

2.3 Building the Recommendation System utilizing the mean of the rating

The initial model (**Naive model with just the mean**)tested assumed that the same rating for all the movies and users following the next equation:

Yui= mu +eiu

mu=true rating of all movies

The following code describe the construction of the estimates and the predictors for model 1:

```
med_hat <- mean(edx_train_set$rating) # rating mean calculation

naive_edx_rmse <- RMSE(med_hat, validationedx$rating) # naive rmse calculation

edx_rmse_results <- data_frame(method = "Naive model with just the mean", RMSE = naive_edx_rmse)
```

Table 1. RMSE results for model 1

method	RMSE
Naive model with just the mean	1.059904

The initial RMSE is a little above 1, therefore it need to be improved.

2.4 Recommendation system utilizing movieID

to improve model 1, The second model (Movie Effect Model) included an average ranking for an specific movie **i**.

Yui = mu +bi+ eiu

mu = true rating of all movies

bi = average ranking for movie i

The following code describe the construction of the estimates and the predictors for model 2:

```
meanedx <- mean(edx_train_set$rating) #mean calculation
movie_med <- edx_train_set %>%
  group_by(movieId) %>%
  summarize(b_im = mean(rating - meanedx)) #mean rating per movieID

# Rating prediction using movieID as unique predictor
predicted_ratings_edx <- meanedx + validationedx %>%
  left_join(movie_med, by='movieId') %>%
  .$b_im

#Model 1 RMSE CALCULATION
model_1_rmse_1 <- RMSE(predicted_ratings_edx, validationedx$rating)
rmse_results_1 <- bind_rows(edx_rmse_results,
                            data_frame(method="Movie Effect Model",
                                       RMSE = model_1_rmse_1 ))
```

Table 2. Comparison of RMSE results from r model 1 and 2

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429

The inclusion of movieId improved the RMSE in **10.9%**; however, this improvement is further from the RMSE values required by the exercise (RMSE < 0.86490).

2.5 Recommendation system utilizing the mean rating produced from movieID and userID

The third model (**Movie + User Effects Model**) tested included an average ranking for an specific movie **i**.

Yui = mu +bi+ bu + eiu

mu = true rating of all movies bi = average ranking for movie i bu = user-specific effect

The following code describe the construction of the estimates and the predictors for model 3:

```
user_med <- edx_train_set%>%
  left_join(movie_med, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_um = mean(rating - meanedx - b_im))

predicted_ratings_2 <- validationedx %>%
  left_join(movie_med, by='movieId') %>%
  left_join(user_med, by='userId') %>%
  mutate(pred = meanedx + b_im + b_um) %>%
  .$pred

model_2_rmse_2 <- RMSE(predicted_ratings_2, validationedx$rating)
```

```
rmse_results_2 <- bind_rows(rmse_results_1,
                            data_frame(method="Movie + User Effects Model",
                                       RMSE = model_2_rmse_2 ))
```

Table 3. Comparison of RMSE results from model 1, 2 and 3

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659319

The inclusion of userId improved the RMSE in **18.3 %** compared with model 1 and **8.2 %** compared with model 2; although, this improvement is getting closer to the values required by the exercise (RMSE $<$ 0.86490) this improvement is not good enough.

2.6 Regularized model using movieID

The fourth model (**Regularized Movie Effect Model**) tested included an the regularization for the estimation of the movie effect

Yui = mu +bi

mu = true rating of all movies bi= average ranking for movie i with penalty using a lambda=3

The following code describe the construction of the estimates and the predictors for model 4 using a lambda of 3:

```
lambda <- 3
movie_reg_med_edx <- edx_train_set %>%
  group_by(movieId) %>%
  summarize(b_im = sum(rating - meanedx)/(n()+lambda), n_i = n())

predicted_ratings_3 <- validationedx %>%
  left_join(movie_reg_med_edx, by='movieId') %>%
  mutate(pred = meanedx + b_im) %>%
  .$pred

model_3_rmse_3 <- RMSE(predicted_ratings_3, validationedx$rating)
rmse_results_3 <- bind_rows(rmse_results_2,
                            data_frame(method="Regularized Movie Effect Model",
                                       RMSE = model_3_rmse_3  ))
```

Table 4. Comparison of RMSE results from model 1, 2, 3 and 4

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659319
Regularized Movie Effect Model	0.9436762

The inclusion of a regularization on movieID improved the RMSE compared with the unregularized model; however, it did not improve the model compared with the model including movieID and userID.

2.7 Regularized Movie + User Effect Model

The fifth model (**Regularized Movie + User Effect Model**) tested included an the regularization for the estimation of the movie effect and the user effect.

Yui = mu + bi + bu

mu = true rating of all movies

bi = average ranking for movie i

bu = user-specific effect

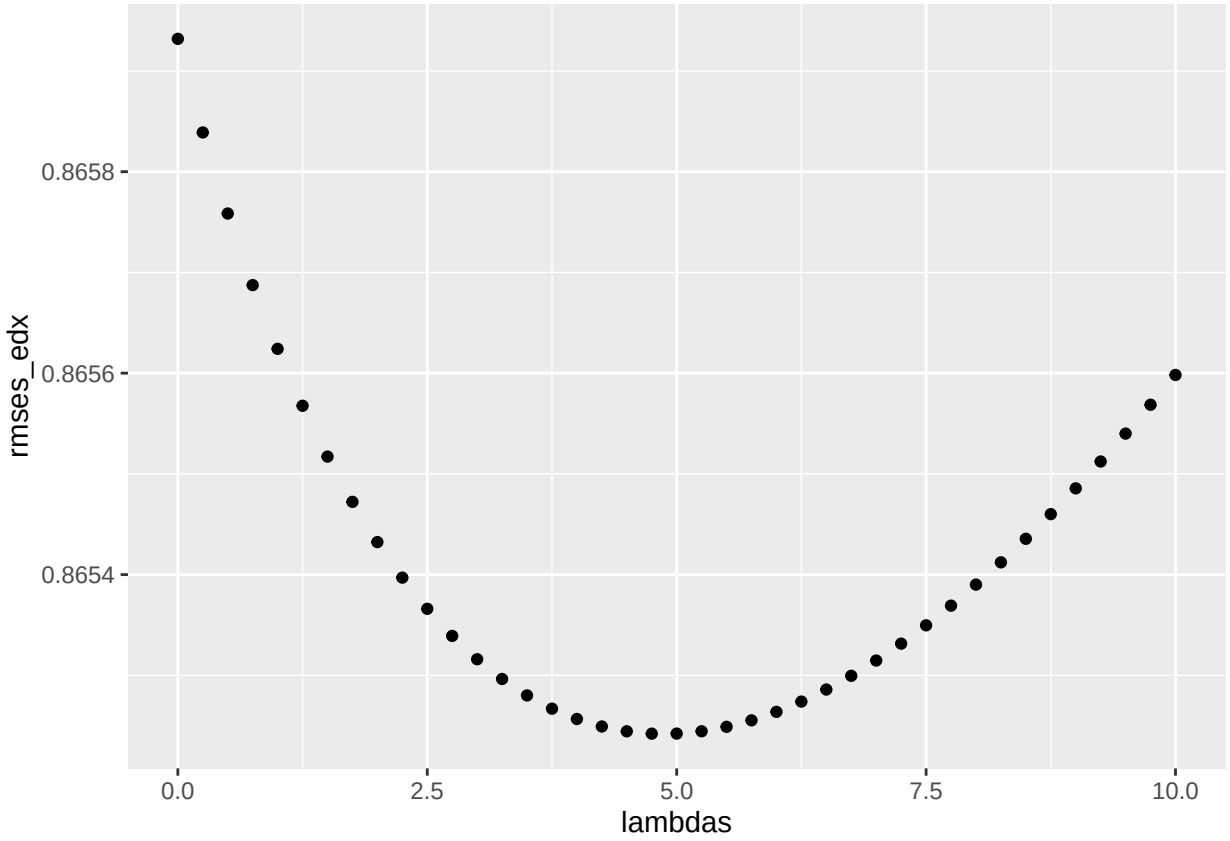
In this case, regularization was utilized to estimate the effect of movieid and user id

The following code describe the construction of the estimates and the predictors for model 5 using crossvalidation to pick lambda.

```
lambdas <- seq(0, 10, 0.25)
rmses_edx <- sapply(lambdas, function(l){
  med <- mean(edx_train_set$rating)
  b_im <- edx_train_set %>%
    group_by(movieId) %>%
    summarize(b_im = sum(rating - med)/(n()+l))
  b_um <- edx_train_set %>%
    left_join(b_im, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_um = sum(rating - b_im - med)/(n()+l))
  predicted_ratings <-
    validationedx %>%
    left_join(b_im, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    mutate(pred = med + b_im + b_um) %>%
    .$pred
  return(RMSE(predicted_ratings, validationedx$rating))
})

lambda_rmses_edx <- data_frame(model= "Regularized Movie + User Effect Model", lambda=lambdas[which.min

rmse_results_4 <- bind_rows(rmse_results_3,
                            data_frame(method="Regularized Movie + User Effect Model",
                                       RMSE = min(rmses_edx)))
```

Figure 1 depict the crossvalidation process used to calculate the lambda that produce the minimum RSME using the predictors from model 5.

Figure 1. lambda vs RMSE plot for model 5



The lambda obtained by the crossvalidation process is described in **Table 5.**

Table 5. lambda obtained from the crossvalidation process for model 5

model	lambda
Regularized Movie + User Effect Model	4.75

Table 6 includes a comparison between the fourth models evaluated in this report.

Table 6. Comparison of RMSE results from model 1, 2, 3, 4 and 5

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659319
Regularized Movie Effect Model	0.9436762
Regularized Movie + User Effect Model	0.8652421

The inclusion of a regularization on movieID and userID improved the RMSE compared with the other models; however, the RSME improved less than **0.0515 %** compared with the unregularized model including movieId and userId and it is still further from the required RSME **(RMSE < 0.86490)**.

2.8 Regularized Movie + User Effect Model + Date (week)

The sixth model (**Regularized Movie + User Effect Model + Date (week)**) tested included the regularization for the estimation of the movie, user and date effect.

Yui = mu + bi + bu + du

mu = true rating of all movies

bi = average ranking for movie i

bu = user-specific effect

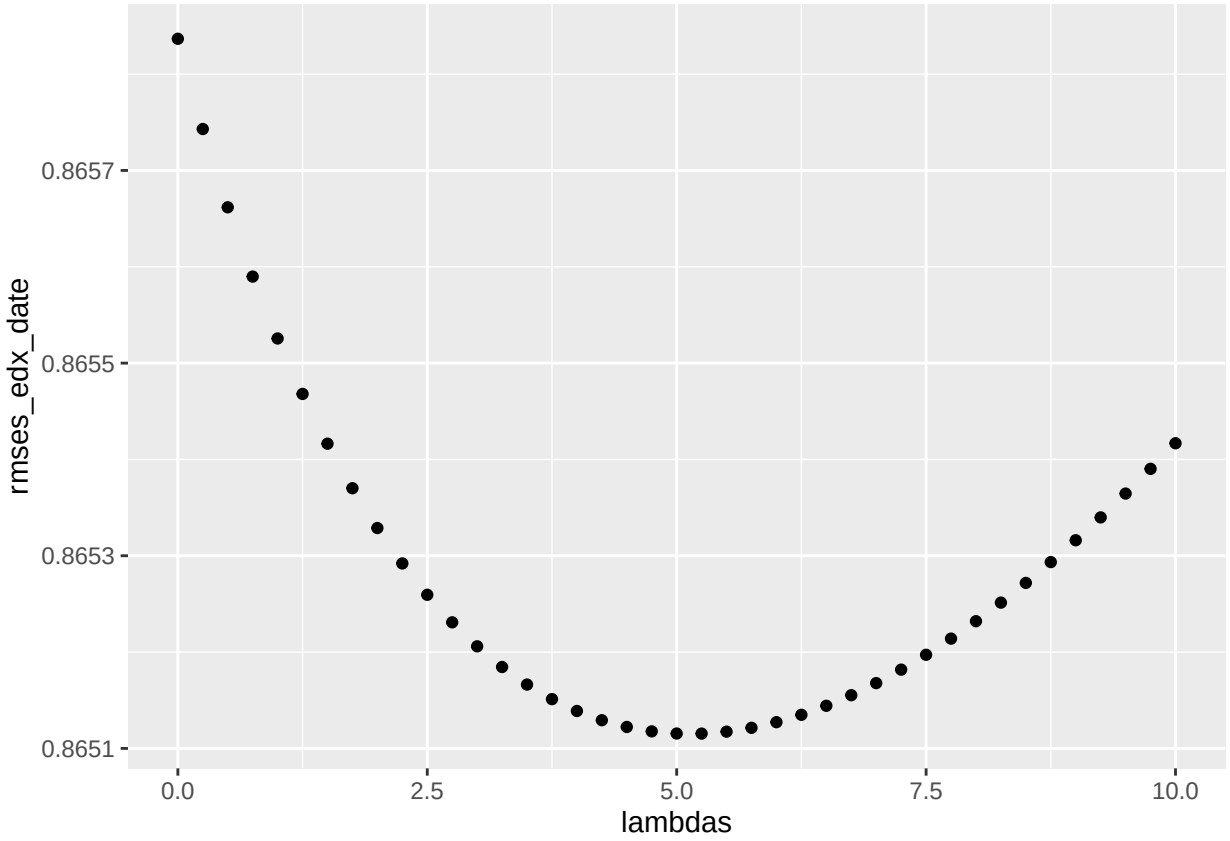
du = date specific effect measured by week

Regularization was utilized to estimate the effect of movieid, user id and data organized by weeks and The following code describe the construction of the estimates and the predictors for model 6 using crossvalidation to pick lambda.

```
edx_train_date <- mutate(edx_train_set, date= as_datetime(timestamp)) #transform time stamp in datetime
edx_test_date <- mutate(validationedx, date= as_datetime(timestamp)) #transform time stamp in datetime

lambdas <- seq(0, 10, 0.25)
rmses_edx_date <- sapply(lambdas, function(l){
  med <- mean(edx_train_date$rating)
  b_im <- edx_train_date %>%
    group_by(movieId) %>%
    summarize(b_im = sum(rating - med)/(n()+l))
  b_um <- edx_train_date %>%
    left_join(b_im, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_um = sum(rating - b_im - med)/(n()+l))
  d_um <- edx_train_date %>%
    left_join(b_im, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    mutate(date = round_date(date, unit = "week")) %>%
    group_by(date) %>% summarize(d_um = sum(rating - b_im - med- b_um)/(n()+l))
  predicted_ratings12 <-
    edx_test_date %>%
    left_join(b_im, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    mutate(date = round_date(date, unit = "week")) %>%
    left_join(d_um, by = "date") %>%
    mutate(pred = med + b_im + b_um+ d_um) %>%
    .$pred
  RMSE(predicted_ratings12, edx_test_date$rating)
})

lambda_rmses_edx_date <- data_frame(model= "Regularized Movie + User Effect + Date(week) Model", lambda=

rmse_results_5 <- bind_rows(rmse_results_4,
                            data_frame(method="Regularized Movie + User Effect + Date(week) Model",
                                       RMSE = min(rmses_edx_date)))
```

Figure 2 depict the crossvalidation process used to calculate the lambda that produce the minimum RSME using the predictors from model 6.

Figure 2. lambda vs RMSE plot for model 6



The lambda obtained by the crossvalidation process is described in **Table 7.**

Table 7. lambda obtained from the crossvalidation process for model 6

model	lambda
Regularized Movie + User Effect + Date(week) Model	5.25

Table 8 includes a comparison between the fourth models evaluated in this report.

Table 8. Comparison of RMSE results from model 1, 2, 3, 4, 5 and 6

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429

method	RMSE
Movie + User Effects Model	0.8659319
Regularized Movie Effect Model	0.9436762
Regularized Movie + User Effect Model	0.8652421
Regularized Movie + User Effect + Date(week) Model	0.8651154

The inclusion of a regularization on a model including movieID, userID, and date(weeks) improved the RMSE compared with the other models; however, the RSME improved less than **0.013 %** compared with the regularized model including movieId and userId and it is still further from the required RSME **(RMSE < 0.86490)** .

2.8 Regularized Movie + User Effect Model + rating date (week)+ genres

The seventh model (**Regularized Movie + User Effect Model + rating date (week)+ genres**) tested included an the regularization for the estimation of the movie, user and date effect.

Yui = mu + bi + bu + du + gu

mu = true rating of all movies

bi = movie especific effect

bu = user-specific effect

du = date specific effect measured by week

gu= genres specific effect

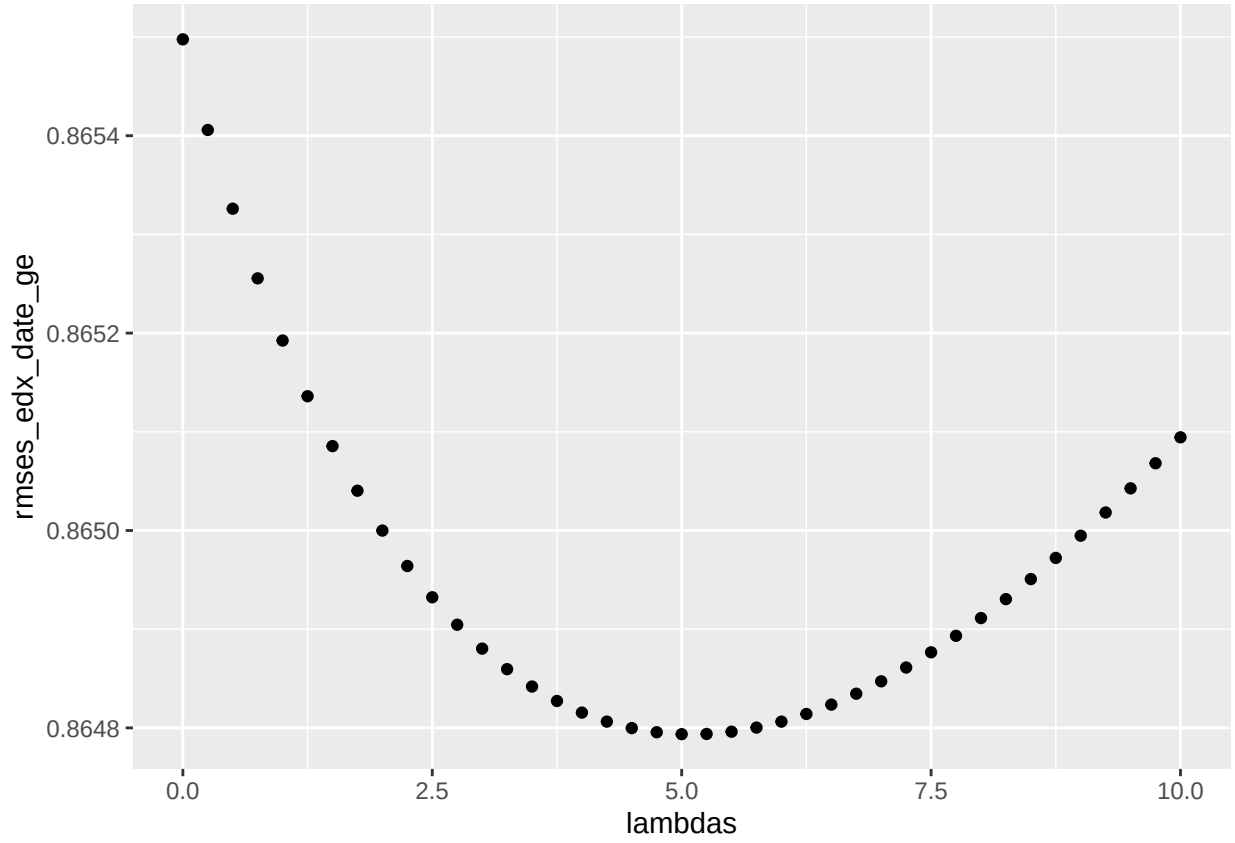
Regularization was utilized to estimate the effect of movieid, user id, data organized by weeks and genres The following code describe the construction of the estimates and the predictors for model 7 using crossvalidation to pick lambda.

```
edx_train_date <- mutate(edx_train_set, date= as_datetime(timestamp)) #transform time stamp in datetime
edx_train_date1 <- mutate(edx_train_date, date = round_date(date, unit = "week"))#transform date in wee

edx_test_date <- mutate(validationedx, date= as_datetime(timestamp)) #transform time stamp in datetime
edx_test_date1 <- mutate(edx_test_date, date = round_date(date, unit = "week"))#transform date in weeks


lambdas <- seq(0, 10, 0.25)
rmses_edx_date_ge <- sapply(lambdas, function(l){
  med <- mean(edx_train_date1$rating)
  b_im <- edx_train_date1 %>%
    group_by(movieId) %>%
    summarize(b_im = sum(rating - med)/(n()+l))
  b_um <- edx_train_date1 %>%
    left_join(b_im, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_um = sum(rating - b_im - med)/(n()+l))
  d_um <- edx_train_date1 %>%
    left_join(b_im, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    group_by(date) %>% summarize(d_um = sum(rating - b_im - med- b_um)/(n()+l))
  g_um <- edx_train_date1 %>%
    left_join(b_im, by = "movieId") %>%
```

```
    left_join(b_um, by = "userId") %>%
    left_join(d_um, by = "date") %>%
    group_by(genres)%>% summarize(g_um = sum(rating - b_im - med- b_um - d_um)/(n()+l))
  predicted_ratings14 <-
    edx_test_date1 %>%
    left_join(b_im, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    left_join(d_um, by = "date") %>%
    left_join(g_um, by = "genres")%>%
    mutate(pred = med + b_im + b_um + d_um + g_um) %>%
    .$pred
  RMSE(predicted_ratings14, edx_test_date1$rating)
})

lambdae_edx_date_ge <- data_frame(model= "Regularized Movie + User Effect + Date(week) + Genre Model", l


rmse_results_6 <- bind_rows(rmse_results_5,
                            data_frame(method="Regularized Movie + User Effect + Date(week) + Genre Mode
                                       RMSE = min(rmses_edx_date_ge)))
```

Figure 3 depict the crossvalidation process used to calculate the lambda that produce the minimum RSME using the predictors from model 7.

Figure 3. lambda vs RMSE plot for model 7



The lambda obtained by the crossvalidation process is described in **Table 9.**

Table 9. lambda obtained from the crossvalidation process for model 7

model	lambda
Regularized Movie + User Effect + Date(week) + Genre Model	5

Table 10 includes a comparison between the fourth models evaluated in this report.

Table 10. Comparison of RMSE results from model 1, 2, 3, 4, 5, 6 and 7

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659319
Regularized Movie Effect Model	0.9436762
Regularized Movie + User Effect Model	0.8652421
Regularized Movie + User Effect + Date(week) Model	0.8651154
Regularized Movie + User Effect + Date(week) + Genre Model	0.8647936

The inclusion of regularization on a model with movieID, userID, date(weeks) and genres improved the RMSE compared with the other models; in this case, the RSME improved **0.038 %** compared with the regularized model including movieId userId and date. Although the improvement was not significant, The RSME obtained by model 7 **(0.86479)** reached the RSME **(RMSE < 0.86490)** level required by the exercise. Therefore this model will be utilized to evaluate the original training **(edx)** and test **(validation)** tests.

3. RESULTS

3.1 VALIDATION OF THE MODEL (Regularized Movie + User Effect Model + rating date (week)+ genres) USING THE ORIGINAL PARTION EDX AND VALIDATION

The first step to evaluate the orginal sets include the transformation of the timestamp into weeks

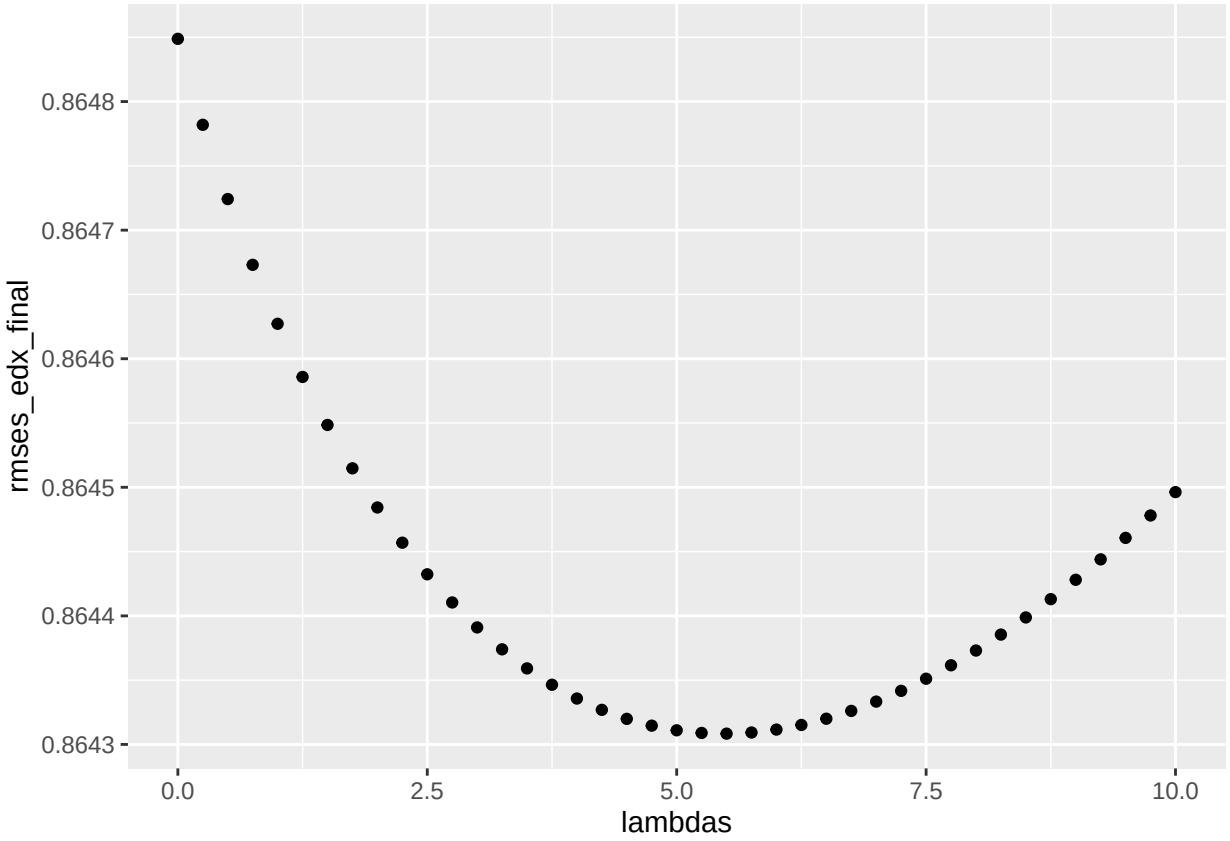
```
library(lubridate)
edx_dates <- mutate(edx, date= as_datetime(timestamp)) #transform time stamp in datetime
edx_dates1 <- mutate(edx_dates, date = round_date(date, unit = "week"))#transform date in weeks

validation_date <- mutate(validation, date= as_datetime(timestamp)) #transform time stamp in datetime
validation_date1 <- mutate(validation_date, date = round_date(date, unit = "week"))#transform date in w
```

```
lambdas <- seq(0, 10, 0.25)
rmses_edx_final<- sapply(lambdas, function(l){
  medfi <- mean(edx_dates1$rating)
  b_imfi <- edx_dates1%>%
```

```
    group_by(movieId) %>%
    summarize(b_imfi = sum(rating - medfi)/(n()+l))
  b_umfi <- edx_dates1 %>%
    left_join(b_imfi, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_umfi = sum(rating - b_imfi - medfi)/(n()+l))
  d_umfi <- edx_dates1 %>%
    left_join(b_imfi, by = "movieId") %>%
    left_join(b_umfi, by = "userId") %>%
    group_by(date) %>% summarize(d_umfi = sum(rating - b_imfi - medfi- b_umfi)/(n()+l))
  g_umfi <- edx_dates1 %>%
    left_join(b_imfi, by = "movieId") %>%
    left_join(b_umfi, by = "userId") %>%
    left_join(d_umfi, by = "date") %>%
    group_by(genres)%>% summarize(g_umfi = sum(rating - b_imfi - medfi- b_umfi - d_umfi)/(n()+l))
  predicted_ratingsf <-
    validation_date1 %>%
    left_join(b_imfi, by = "movieId") %>%
    left_join(b_umfi, by = "userId") %>%
    left_join(d_umfi, by = "date") %>%
    left_join(g_umfi, by = "genres")%>%
    mutate(pred = medfi + b_imfi + b_umfi + d_umfi + g_umfi) %>%
    .$pred
  RMSE(predicted_ratingsf, validation_date1$rating)
})



lambda_final<-  data_frame(model= "Regularized Movie + User + Date + Genre, Validation data", lambda=la

RMSE_Final <- data_frame(model= "Regularized Movie + User + Date + Genre, Validation data", RMSE=min(rm

rmse_results_final <- bind_rows(rmse_results_6,
                            data_frame(method="Regularized Movie + User + Date + Genre, Validation data
                                       RMSE = min(rmses_edx_final)))
```

Figure 4 depicts the crossvalidation process used to calculate the lambda that produce the minimum RSME using the predictors previously selected.

Figure 4. lambda vs RMSE plot for model 7



The lambda obtained by the crossvalidation process is described in **Table 9.**

Table 11. lambda obtained from the crossvalidation process for the RSME using the validation data

model	lambda
Regularized Movie + User + Date + Genre, Validation data	5.5

The reported RSME obtained using the validation set is included in **table 12**.

Table 12. RSEM obtained from the proposed model using the validation data

model	RMSE
Regularized Movie + User + Date + Genre, Validation data	0.8643084

Table 13 includes a comparison between the fourth models evaluated in this report.

Table 13. Comparison of RMSE results using the validation data and the models from the previous section

method	RMSE
Naive model with just the mean	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8659319
Regularized Movie Effect Model	0.9436762
Regularized Movie + User Effect Model	0.8652421
Regularized Movie + User Effect + Date(week) Model	0.8651154
Regularized Movie + User Effect + Date(week) + Genre Model	0.8647936
Regularized Movie + User + Date + Genre, Validation data	0.8643084

The final RMSE obtained using the validation set **(0.86431)** had the lowest RSME value if compared with the previous tested models it had a better performance this is expected as the number of data increased. Additionally, the value obtained is below the value required by the exercise of **<0.86490**.

CONCLUSION

The recommendation system developed in this report is a fair approximation to predict movies ratings based on the regularized user, movie, genre and dates predictors. The inclusion of these variables improved the recommendation system compared with models with less number of variables; additionally, the use of regularization improved the bias that could be in the data if this process is not included. The recommendation system produced a RMSE below **(0.86431)** the value required by the exercise **<0.86490**. Future approximations can include other predictors or improve the calculations by using dimensional reduction methods.