MovieLens Project Report

Marcus Schmidt

17/02/2021

(I) INTRODUCTION

(I a) Data set & goal

This project aims to predict 1 Million ratings of movies on Netflix based on a machine learning algorithm that uses 9 Million ratings to configure this algorithm. The author used a linear model structure of user, movie, time and genre effect together with regularization and a limit cut to predict ratings. A final rooted mean square error (RMSE) of 0.8646349 was achieved.

(I b) Data set download

The data set can be downloaded and combined using the following code:

```
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))

# if using R 3.6 or earlier, instead use:
# 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")
```

(I c) Data set structure

Here is a peak into the data set, showing its dimension (rows x columns), variable names and data structure. We can see here that user and movie are numbers. We will later turn them into factors.

```
dim(movielens) # dimension of dataset
```

```
## [1] 10000054        6
```

```
names(movielens) # variable names
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

```
as_tibble(head((movielens))) # see variable types
```

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title                   genres
##    <int>   <dbl>  <dbl>     <int> <chr>                   <chr>
## 1      1     122      5 838985046 Boomerang (1992)        Comedy|Romance
## 2      1     185      5 838983525 Net, The (1995)         Action|Crime|Thriller
## 3      1     231      5 838983392 Dumb & Dumber (1994)    Comedy
## 4      1     292      5 838983421 Outbreak (1995)         Action|Drama|Sci-Fi|T~
## 5      1     316      5 838983392 Stargate (1994)         Action|Adventure|Sci-~
## 6      1     329      5 838983392 Star Trek: Generations~ Action|Adventure|Dram~
```

(II) METHODS & ANALYSIS

(II a) Variable factorization

I am factorizing movieId and userId for the linear models. I also add a variable "week", since there might also be a temporal effect in rating movies.

```
movielens <- movielens %>% mutate(movieId = as.factor(movieId), userId = as.factor(userId),
                                  week = round_date(as_datetime(timestamp), "week"))
str(movielens$week)
```

```
##  POSIXct[1:10000054], format: "1996-08-04" "1996-08-04" "1996-08-04" "1996-08-04" "1996-08-04" ...
```

(II b) Validation set

10% of the data set are held back for final evaluation. This should only include movies, users, weeks and genres that are also in the edx set (which will later be divided into training and test set). Note: Including the "week" and "genre" criteria here does not remove any move observations from the validation set than done with only "movieId" and "userId". The validation data set still has all 999.999 obervations.

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

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

(II c) Data set division into train and test set

When dividing into test and training set, I used 10% of the data set for test purposes, so I have a majority to train the algorithm. This proved to be important since for a long time, I used p = 0.5 which produced a higher RMSE.

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
```

Now I continue building training and test set, but to make sure that the variables in my test set are also in my training set.

```
edx_train <- edx[-test_index,]
temp <- edx[test_index,]

# Make sure userId and movieId and week and genre in test set are also in train set
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")%>%
  semi_join(edx_train, by = "week") %>%
  semi_join(edx_train, by = "genres")

dim(edx_test)
```

```
## [1] 899990      7
```

```
# Add rows removed from test set back into train
removed <- anti_join(temp, edx_test)
```

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

```
dim(removed)
```

```
## [1] 17  7
```

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

Here I check whether the dimensions represent my data splitting:

```
dim(edx_train)
```

```
## [1] 8100065       7
```

```
dim(edx_test)
```

```
## [1] 899990      7
```

(II d) RMSE function

A rooted mean square error (RMSE) function is used to evaluate the performance of the models that will be set up.

```
RMSE <- function(ratings, predictions){
  sqrt(mean((ratings - predictions)^2))
}
```

(II e) Reference model

Next, as reference for all following models, a reference model is created where I simply guess the rating as the overall mean rating of the test set:

```r
mu <- mean(edx_train$rating)
mu # 3.51
```

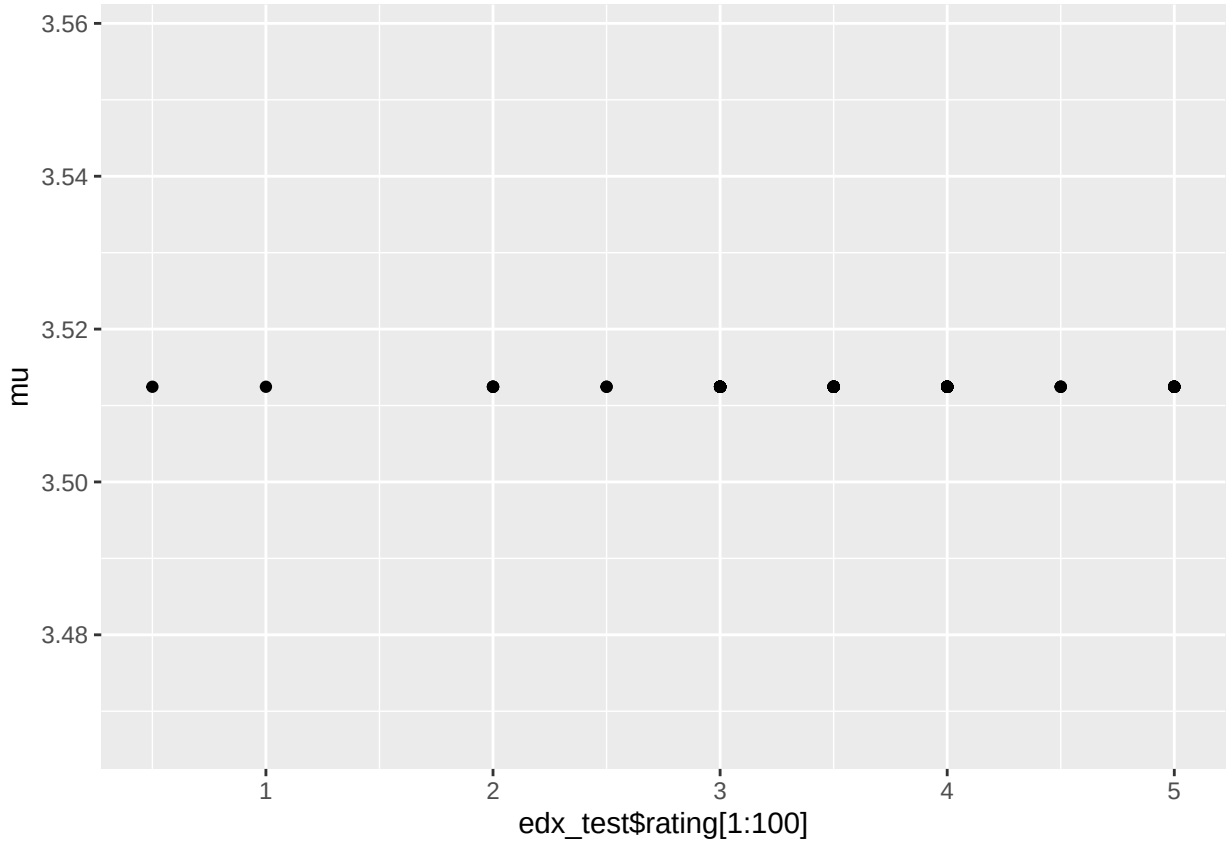
```
## [1] 3.512456
```

```r
# calculating RMSE for reference model
rmse_guessing <- RMSE(edx_test$rating, mu)
rmse_guessing
```

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

This is what a plot of 100 predictions looks like for the reference model and a table, which all future RMSEs will be listed in:

```r
# plotting 100 predictions for reference model
qplot(edx_test$rating[1:100], mu)
```




```r
# building a RMSE table to see model performances
rmse_table <- data.frame(model = "guessing", rmse = rmse_guessing)
rmse_table
```

```
##      model     rmse
## 1 guessing 1.060054
```

(II f) Linear models

Linear models can usually be run using the lm() function. However, in a large data set like this, it is faster to add the group effect to the mean. I will show how this was done for movie and user effects.

Linear model with movie effect

In order to get an effect on the overall mean rating of each individual movie, one has to summarize the training data by movie, get a mean and then add this mean to each observation in the test set.

```
#### Movie effect ####

# create effect-variable = the deviation fromt he overall mean for this effect
movie_means <- edx_train %>% group_by(movieId) %>% summarize(movie_effect = mean(rating - mu))
head(movie_means)

# hist(movie_means$movie_effect) # visualize distribution of the effect

pred1 <- mu + # adding the effect to the mean
  edx_test %>% left_join(movie_means, by = "movieId") %>% .$movie_effect # .$ equals pull()

# calculating RMSE
rmse_1 <- RMSE(edx_test$rating, pred1)
rmse_1

# adding effect to the table
rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie effect", rmse = rmse_1))
rmse_table
```

Linear model with movie + user effect

This is similar to the first linear model, but here, the movie effect is first taken into account before adding the user effect.

```
#### Movie & user effect ####

# create effect variable, note that this is on top of the movie effect
user_means <- edx_train %>% left_join(movie_means, by = "movieId") %>%
  group_by(userId) %>% summarize(user_effect = mean(rating - mu - movie_effect))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(user_means)
```

```
## # A tibble: 6 x 2
##   userId user_effect
##   <fct>        <dbl>
## 1 1            1.67
## 2 2           -0.434
## 3 3            0.268
```

```
## 4 4            0.698
## 5 5            0.100
## 6 6            0.336
```

```
pred2 <- mu + # adding the effects to the mean
  edx_test %>% left_join(movie_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_means, by = "userId") %>% .$user_effect

# calculating RMSE
rmse_2 <- RMSE(edx_test$rating, pred2)
rmse_2
```

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

```
# adding effect to the table
rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user effect", rmse = rmse_2))
rmse_table
```

```
##                    model      rmse
## 1              guessing 1.0600537
## 2       lm movie effect 0.9429615
## 3 lm movie + user effect 0.8646843
```

```
# plotting 1000 predictions
# qplot(edx_test$rating[1:1000], pred2[1:1000])
```

Further proceedings: including "week" and "genre"

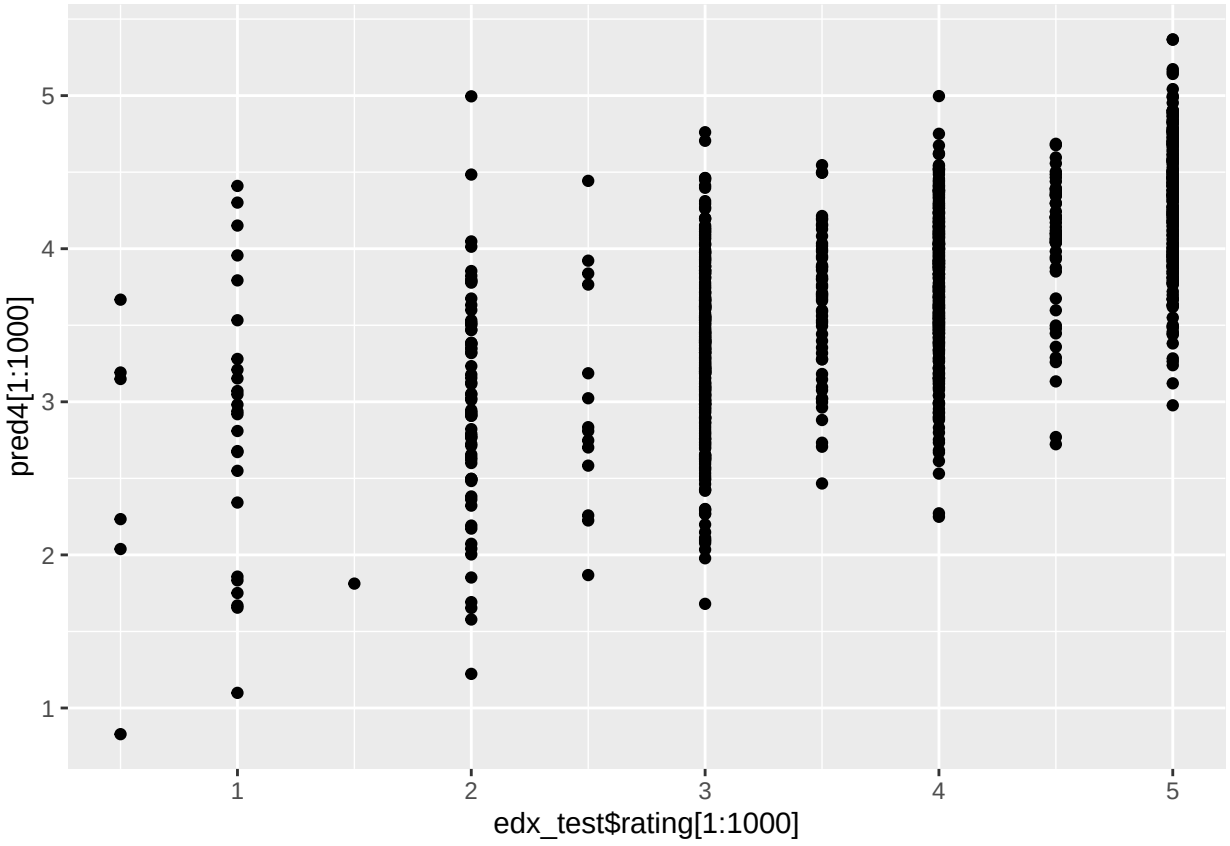
This was further done the same way with the "week" and the "genre" variable. The long, full code can be seen in Appendix 1 to keep the main report tidy and readable.

Here are the results achieved by the first four linear models and a plot of the fourth one. The first 1000 predictions were plotted:

```
rmse_table
```

```
##                                  model      rmse
## 1                            guessing 1.0600537
## 2                     lm movie effect 0.9429615
## 3                lm movie + user effect 0.8646843
## 4         lm movie + user + time effect 0.8645933
## 5 lm movie + user + time + genre effect 0.8642321
```

```
# plotting 1000 predictions
qplot(edx_test$rating[1:1000], pred4[1:1000])
```



(II g) Linear models with regulatization

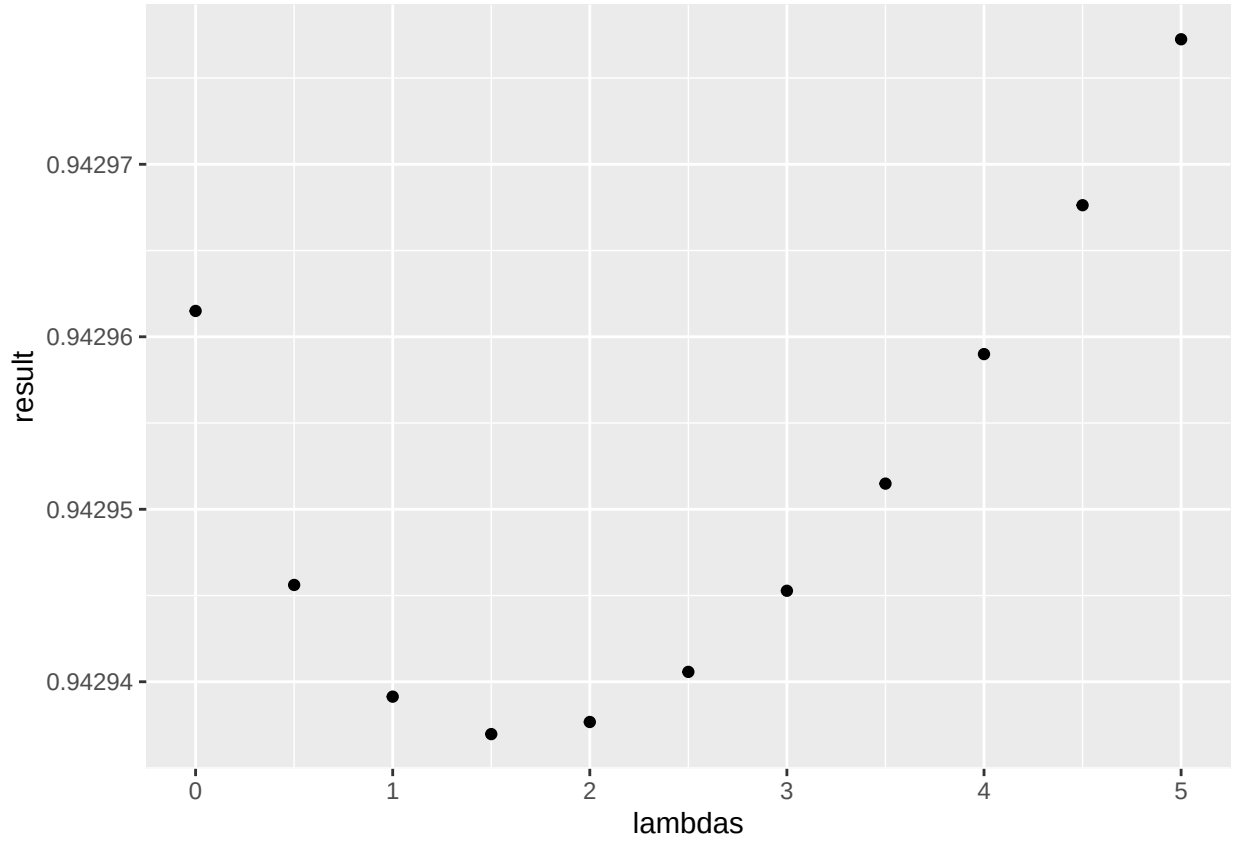
Regulatization is know to often improve predictions because group with few observations may not represent the true deviation of that group from the overall mean. Adding a penalty term lambda reduces the effect of a small group. In the following, I run through a range of lambdas with sapply() and then select the best lamda, which is then used to calculate the effect. I show this for movie and user.

Linear model with movie effect and regulatization

In this first regularized model, we see that instead of taking the mean for each movie, the sum is divided not by the number of observations, but by the number plus a penalty term (lambda). When the number of observations is high (in this example, a movie is watched by many), the penalty term matters less and less.

```
#### Creating linear models with regularization ####
# note: regularization reduces the effect of categories that appear less often
# through dividing through a higher number when taking the mean effect of that category

#### Movie effect with regularization ####

# choosing lambda, the penalty number
lambdas <- seq(0,5, by = 0.5)

# running trough a range of lambdas
result <- sapply(lambdas, function(lambda){
```

```
  movie_reg_means <- edx_train %>%
    group_by(movieId) %>%
    summarize(movie_effect = sum(rating - mu)/(n()+lambda), n_movies = n()) # here, lambda is added

  pred_1r <- mu +
    edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect

  rmse_1r <- RMSE(edx_test$rating, pred_1r)
  rmse_1r
})

# plotting lambdas and resulting RMSE
qplot(lambdas, result)
```



Note:

In the plot above, we see the lambdas that are tried (x-axis) and how this affects the RMSE (y-axis). In the further evaluation, the best lamda is taken to calculate the group effect.

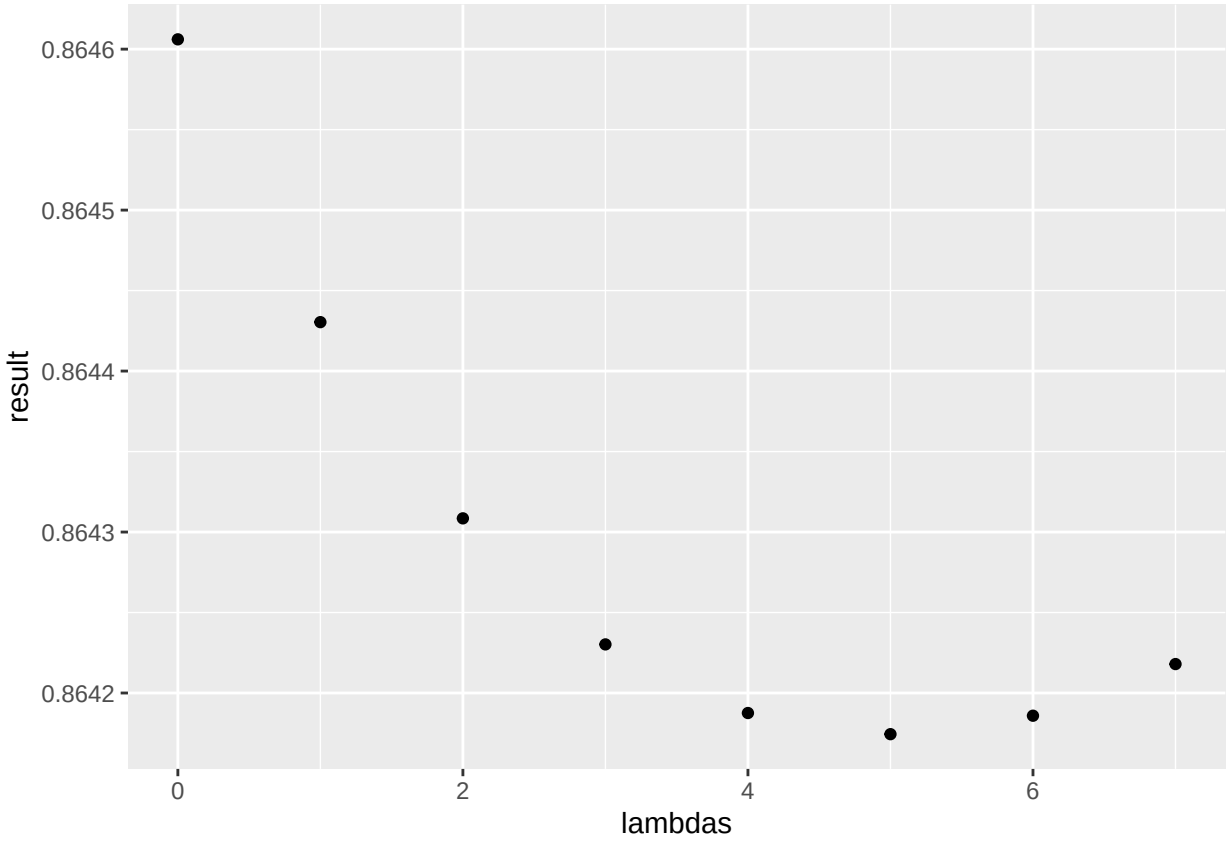
```
# choosing best lambda
best_lambda <- lambdas[which.min(result)]
best_lambda

# from here, the chosen lambda is taken to calculate the regularized effect
```

```
lambda = best_lambda

movie_reg_means <- edx_train %>%
  group_by(movieId) %>%
  summarize(movie_effect = sum(rating - mu)/(n()+lambda), n_movies = n()) # here, lamda is added

head(movie_reg_means)

pred_1r <- mu +
  edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect

rmse_1r <- RMSE(edx_test$rating, pred_1r)
rmse_1r

rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie effect (with regularization)", rmse = rmse_1r))
rmse_table

# plotting 1000 predictions
# qplot(edx_test$rating[1:1000], pred_1r[1:1000])
```

Linear model with movie + user effect and regulatization

This is similar to the first regularized model, but now the regularized user effect is added.

```
#### Movie + user effect with regulatization ####

# choosing lambda, the penalty number
lambdas <- seq(0,7, by = 1)

result <- sapply(lambdas, function(lambda){
  user_reg_means <- edx_train %>%
    left_join(movie_reg_means, by = "movieId") %>%
    group_by(userId) %>%
    summarize(user_effect = sum(rating - mu - movie_effect)/(n()+lambda), n_user = n())

  pred_2r <- mu +
    edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
    edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect

  rmse_2r <- RMSE(edx_test$rating, pred_2r)
  rmse_2r
})

# plotting lambdas and resulting RMSE
qplot(lambdas, result)
```




```r
# choosing best lambda
best_lambda <- lambdas[which.min(result)]
best_lambda


#from here, the chosen lambda is taken to calculate the regularized effect
lambda = best_lambda

user_reg_means <- edx_train %>%
  left_join(movie_reg_means, by = "movieId") %>%
  group_by(userId) %>%
  summarize(user_effect = sum(rating - mu - movie_effect)/(n()+lambda), n_user = n())
head(user_means)

pred_2r <- mu +
  edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect

rmse_2r <- RMSE(edx_test$rating, pred_2r)
rmse_2r

rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user effect (with regulatization)", rmse = rmse_2r))
rmse_table

# plotting 1000 predictions
```

```
# qplot(edx_test$rating[1:1000], pred_2r[1:1000])
```

Further proceedings: including "week" and "genre" with regulatization

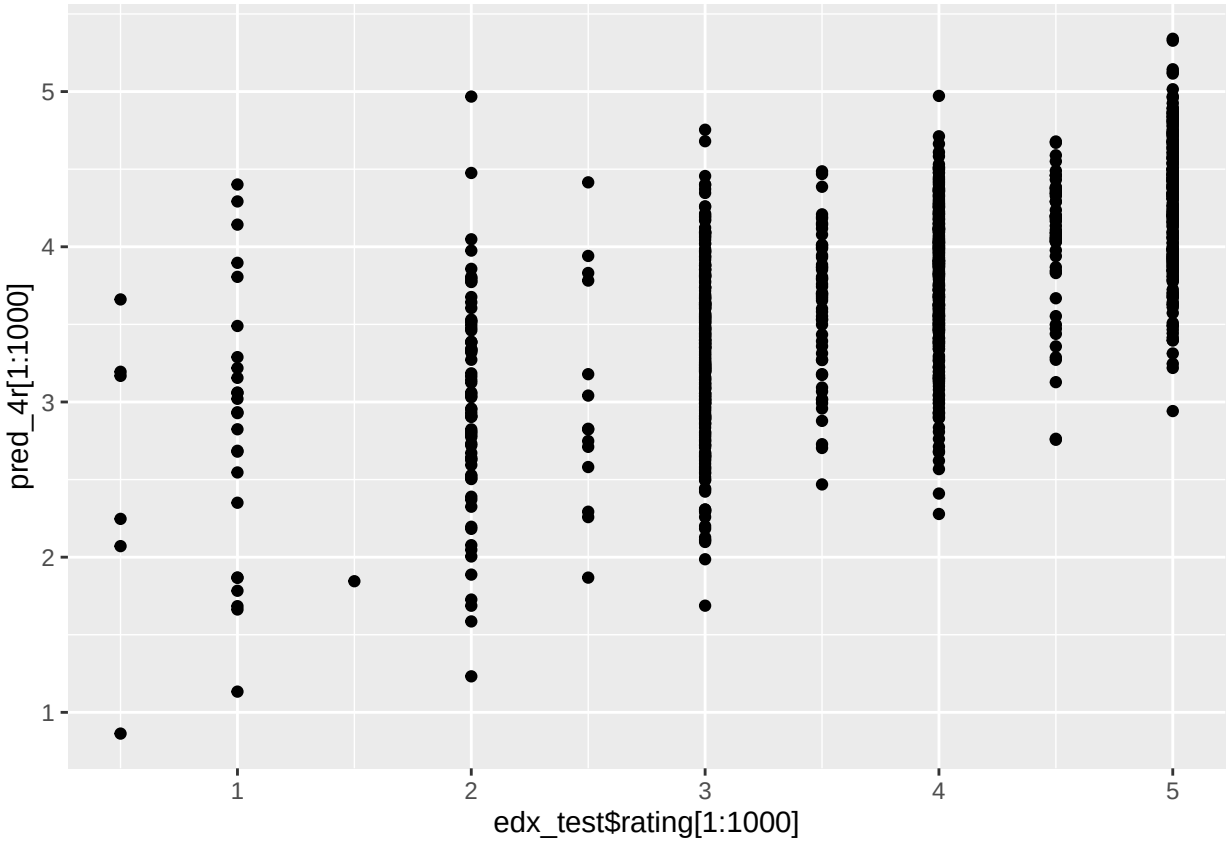
This regulatization procedure was further done the same way with the "week" and "genre" variables and regulatization in the same way. The long, full code can be seen in Appendix 2 to keep the main report tidy and readable.

Here are the results of the four regularized linear models and a plot of the first 1000 predictions:

```
rmse_table[6:9,]
```

```
##                                                    model      rmse
## 6                      lm movie effect (with regularization) 0.9429370
## 7                lm movie + user effect (with regulatization) 0.8641744
## 8         lm movie + user + time effect (with regulatization) 0.8640574
## 9 lm movie + user + time + genre effect (with regularization) 0.8637170
```

```
# plotting 1000 predictions
qplot(edx_test$rating[1:1000], pred_4r[1:1000])
```



(II h) Limit cutting

Some predicted ratings are lower than 1 and higher than 5. This is why those will be assigend to 1 and 5 respectively.

```
#### adding a limit cut ####

# check whether there are predictions which are below 1 or higher than 5 which they cannot be
range(pred_4r)
```

```
## [1] -0.5170148  5.8631832
```

```
# letting a function run over the predictions and setting
# those that are lower than 1 to 1 and
# those that are higher than 5 to 5
pred_4r_limcut <-
  sapply(pred_4r, function(x){
    if(x < 1){x <- 1}
    if(x > 5){x <- 5}
    x
  })
```

This also improved the RMSE:

```
rmse_4r_limcut
```

```
## [1] 0.8635486
```

(III) RESULTS

(III a) RMSEs of tested models

Here is an overview off all tested models and their performance:

```
rmse_table
```

```
##                                                      model      rmse
## 1                                                guessing 1.0600537
## 2                                            lm movie effect 0.9429615
## 3                                      lm movie + user effect 0.8646843
## 4                                lm movie + user + time effect 0.8645933
## 5                          lm movie + user + time + genre effect 0.8642321
## 6                          lm movie effect (with regularization) 0.9429370
## 7                  lm movie + user effect (with regulatization) 0.8641744
## 8           lm movie + user + time effect (with regulatization) 0.8640574
## 9  lm movie + user + time + genre effect (with regularization) 0.8637170
## 10                          lm 4 regularized effects + limit cut 0.8635486
```

(III b) Evaluation of best model on validation set

I now create the final prediction from the validation set. It includes the regularized linear effects of movie, user, time and genre as well as a limit cut for values over 5 or under 1, which are assigned 5 and 1, respectively.

```
pred_final <- (
  mu +
  validation %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
  validation %>% left_join(user_reg_means, by = "userId") %>% .$user_effect +
  validation %>% left_join(time_reg_means, by = "week") %>% .$time_effect +
  validation %>% left_join(genre_reg_means, by = "genres") %>% .$genre_effect
  ) %>%
  sapply(function(x){
    if(x < 1){x <- 1}
    if(x > 5){x <- 5}
    x
  })

# check a few values to see if they make sense:
head(pred_final)
```

```
## [1] 4.243883 4.955008 4.343018 3.231767 4.136730 2.631547
```

```
# calculating the final RMSE
rmse_final <- RMSE(validation$rating, pred_final)
rmse_final
```

It is now time for output of the final RMSE (drums, please :-)):

```
## [1] 0.8646349
```

(IV) CONCLUSION

(IV a) Summary of the report

It proved to be important to use a large part of the data set for training rather than splitting training and test set in half. Furthermore, the variables of movie and user were more important in reducing the RMSE than the time or the genre. Regulatization also reduced the RMSE, underlining that small groups are harder to predict and thus should be given smaller importance. Sind prediction yielded values out of the existing range from 1 to 5 starts, cutting those predictions to a minimum of 1 and a maximum of 5 also reduced the RMSE. To sum up, many small steps to improve a model will in the end yield significant improvements.

(IV b) Limitations

Even more powerful methods such as random forest, knn or the recommenderlab package which uses matrix factorization exceeded the calculation capacity of a regular computer or laptop. Furthermore, it appears to me that using knn can be powerful but since it creates n-dimensional coordinate systems, numerical variables may be better suited. However, linear models with the twist of regulatization and a final limit cut achieved a satisfying result.

(IV c) Final thought

I have noticed that Netflix has turned from the star rating to a binary question of did you like the movie or not. My assumption (and I will research on this after submitting the report), is that this may yield a better prediction by being able to use decision rather than regression based machine learning algorithms. Potentially, more factor variables yield a better decision while more numerical variables yield a better regression. That is something I aim to test in further machine learning projects.

(X) APPENDIX

Appendix 1 - Code for linear models including time and genre

```
#### Movie & user & time effect ####

# create effect variable, note that this is on top of the movie & user effects
time_means <- edx_train %>%
  left_join(movie_means, by = "movieId") %>%
  left_join(user_means, by = "userId") %>%
  group_by(week) %>% summarize(time_effect = mean(rating - mu - movie_effect - user_effect))
head(time_means)

pred3 <- mu + # adding the effects to the mean
  edx_test %>% left_join(movie_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_means, by = "userId") %>% .$user_effect +
  edx_test %>% left_join(time_means, by = "week") %>% .$time_effect

# calculating RMSE
rmse_3 <- RMSE(edx_test$rating, pred3)
rmse_3

# adding effect to the table
rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user + time effect", rmse = rmse_3))
rmse_table

# plotting 1000 predictions
qplot(edx_test$rating[1:1000], pred3[1:1000])



#### Movie & user & time & genre effect ####

# create effect variable, note that this is on top of the movie & user & time effects
genre_means <- edx_train %>%
  left_join(movie_means, by = "movieId") %>%
  left_join(user_means, by = "userId") %>%
  left_join(time_means, by = "week") %>%
  group_by(genres) %>% summarize(genre_effect = mean(rating - mu - movie_effect -
                                                       user_effect - time_effect))
head(genre_means)

pred4 <- mu + # adding the effects to the mean
  edx_test %>% left_join(movie_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_means, by = "userId") %>% .$user_effect +
  edx_test %>% left_join(time_means, by = "week") %>% .$time_effect +
  edx_test %>% left_join(genre_means, by = "genres") %>% .$genre_effect

# calculating RMSE
rmse_4 <- RMSE(edx_test$rating, pred4)
rmse_4
```

```r
# adding the effect to the table
rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user + time + genre effect", rmse = rmse_4))
```

Appendix 2 - Code for linear models including time and genre with regularization

```
#### Movie + user + time effect with regularization ####

#choosing lambda, the penalty number
lambdas <- c(0,3,6,7,8,9,10,15,20)

result <- sapply(lambdas, function(lambda){
  time_reg_means <- edx_train %>%
    left_join(movie_reg_means, by = "movieId") %>%
    left_join(user_reg_means, by = "userId") %>%
    group_by(week) %>% summarize(time_effect = sum(rating - mu -
  head(time_reg_means)

  pred_3r <- mu +
    edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
    edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect +
    edx_test %>% left_join(time_reg_means, by = "week") %>% .$time_effect

  rmse_movie_user_time_reg_effect <- RMSE(edx_test$rating, pred_3r)
  rmse_movie_user_time_reg_effect
})

# plotting lambdas and resulting RMSE
qplot(lambdas, result)

# choosing best lambda
best_lambda <- lambdas[which.min(result)]
best_lambda



#from here, the chosen lambda is taken to calculate the regularized effect
lambda = best_lambda

time_reg_means <- edx_train %>%
  left_join(movie_reg_means, by = "movieId") %>%
  left_join(user_reg_means, by = "userId") %>%
  group_by(week) %>% summarize(time_effect = sum(rating - mu -
                    movie_effect - user_effect)/(n()+lambda), n_time = n())
head(time_reg_means)

pred_3r <- mu +
  edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect +
  edx_test %>% left_join(time_reg_means, by = "week") %>% .$time_effect

rmse_3r <- RMSE(edx_test$rating, pred_3r)
rmse_3r

rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user + time effect (with regulatization)",
                               rmse = rmse_3r))
```

```r
rmse_table

# plotting 1000 predictions
qplot(edx_test$rating[1:1000], pred_3r[1:1000])



#### Movie + user + time + genre effect with regulatization ####

# choose lambda, the penalty number
lambdas <- seq(0, 5, by = 1)

result <- sapply(lambdas, function(lambda){
  genre_reg_means <- edx_train %>%
  left_join(movie_reg_means, by = "movieId") %>%
  left_join(user_reg_means, by = "userId") %>%
  left_join(time_reg_means, by = "week") %>%
  group_by(genres) %>% summarize(genre_effect = sum(rating - mu -
                        movie_effect - user_effect - time_effect)/(n()+lambda), n_genre = n())

pred_4r <- mu +
  edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
  edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect +
  edx_test %>% left_join(time_reg_means, by = "week") %>% .$time_effect +
  edx_test %>% left_join(genre_reg_means, by = "genres") %>% .$genre_effect

rmse_movie_user_time_genre_reg_effect <- RMSE(edx_test$rating, pred_4r)
rmse_movie_user_time_genre_reg_effect
})

# plotting lambdas and resulting RMSE
qplot(lambdas, result)

# choosing best lambda
best_lambda <- lambdas[which.min(result)]
best_lambda

# best lambda is 0 here so it does not improve the result


# from here, the chosen lambda is taken to calculate the regularized effect
lambda = best_lambda

genre_reg_means <- edx_train %>%
  left_join(movie_reg_means, by = "movieId") %>%
  left_join(user_reg_means, by = "userId") %>%
  left_join(time_reg_means, by = "week") %>%
  group_by(genres) %>% summarize(genre_effect = sum(rating - mu -
                        movie_effect - user_effect - time_effect)/(n()+lambda), n_genre = n())
head(genre_reg_means)

pred_4r <- mu +
  edx_test %>% left_join(movie_reg_means, by = "movieId") %>% .$movie_effect +
```

```
  edx_test %>% left_join(user_reg_means, by = "userId") %>% .$user_effect +
  edx_test %>% left_join(time_reg_means, by = "week") %>% .$time_effect +
  edx_test %>% left_join(genre_reg_means, by = "genres") %>% .$genre_effect

rmse_4r <- RMSE(edx_test$rating, pred_4r)
rmse_4r

rmse_table <- rbind(rmse_table,
                    data.frame(model = "lm movie + user + time + genre effect (with regularization)",
                               rmse = rmse_4r))
```