MovieLens Project 2020

Christopher James Page

7/20/2020

Introduction/Overview/Executive Summary

The purpose of the MovieLens Project 2020 effort was to develop an algorithm capable of predicting movie ratings with a commercially viable level of accuracy, quantified as the Root Means Square Error (RMSE). Using EDX, a data set constructed by a random sampling of ninety percent of the MovieLens data available at http://files.grouplens.org/datasets/movielens/ml-10m.zip, the author assembled and evaluated nineteen algorithms before selecting the one that generated the lowest RMSE when applied to the EDX test set. That algorithm accounted for the regularized effects of three predictors: (a) the movie being rated, (b) the user registering the rating, and (c) the month in which the rating was registered. It generated an RMSE of less than 0.884 against the EDX test set. The expectation is that it will generate a similar RMSE against a VALIDATION set formed from the remaining ten percent of the original MovieLens data.

Method/Analysis

The first step was to load the necessary libraries from the repository http://cran.us.r-project.org. Those libraries included not only the prescribed tidyverse, caret, and data.table but lubridate. Lubridate was needed in order to make effective use of the timestamp data included in MovieLens and its EDX and VALIDATE extracts.

The second step entailed loading the MovieLens set from http://files.grouplens.org/datasets/movielens/ml-10m.zip and then wrangling the data in a manner that would result in ready access to information detailing the unique identifiers, titles, and genres of the individual movies; the unique identifiers of the people who had registered ratings; and the times when those people registered each of their ratings.

The third step was to generate the EDX and VALIDATION data sets. The EDX data set was used to develop the movie rating prediction algorithms. It was separated into TRAIN and TEST data sets. The VALIDATION data set was set-aside to evaluate the RMSE of the final algorithm. RMSE is the typical error made we make when predicting a movie rating.

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

The next set of steps involved examining and assessing the following nineteen algorithms:

Algorithm 01 (Average Rating). The first algorithm made predictions based solely on the EDX training set's average movie rating.

```
mu <- mean(edx_train_set$rating)

predicted_ratings_algorithm_01 <- mu

RMSE01 <- RMSE(predicted_ratings_algorithm_01, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an unacceptably high RMSE of

```
## [1] 1.060428
```

Algorithm 02 (Average Rating plus Movie Effects). The second algorithm made predictions based on the EDX training set's average movie rating plus the effects of the movies being rated.

```
mu <- mean(edx_train_set$rating)

movie_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu))
```

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

```
predicted_ratings_algorithm_02 <- mu + edx_test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     pull(b_i)

RMSE02 <- RMSE(predicted_ratings_algorithm_02, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded a better but still too high RMSE of

```
## [1] 0.9433553
```

Algorithm 03 (Average Rating plus Regularized Movie Effects). The third algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the movies being rated.

```
lambdas <- seq(0, 30, 0.25)

mu <- mean(edx_train_set$rating)

movie_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(s = sum(rating - mu), n_i = n())
```

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

```
rmses <- sapply(lambdas, function(l){
     predicted_ratings <- edx_test_set %>%
         left_join(movie_avgs, by='movieId') %>%
         mutate(b_i = s/(n_i+l)) %>%
         mutate(pred = mu + b_i) %>%
         pull(pred)
     return(RMSE(predicted_ratings, edx_test_set$rating))
 })
lambda03 <- lambdas[which.min(rmses)]

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```r
predicted_ratings_algorithm_03 <- edx_test_set %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     mutate(pred = mu + b_i) %>%
     pull(pred)

RMSE03 <- RMSE(predicted_ratings_algorithm_03, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded a slightly improved RMSE of

```
## [1] 0.9432842
```

Algorithm 4 (Averate Rating plus User Effects). The fourth algorithm made predictions based on the EDX training set's average movie rating plus the effects of the users registering the ratings.

```r
mu <- mean(edx_train_set$rating)

user_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu))
```

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

```r
predicted_ratings_algorithm_04 <- mu + edx_test_set %>%
    left_join(user_avgs, by='userId') %>%
    pull(b_u)

RMSE04 <- RMSE(predicted_ratings_algorithm_04, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.979119
```

Algorithm 5 (Average Rating plus Regularized User Effects). The fifth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the users registering the ratings.

```r
lambdas <- seq(0, 30, 0.25)

mu <- mean(edx_train_set$rating)

user_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(s = sum(rating - mu), n_i = n())
```

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

```r
rmses <- sapply(lambdas, function(l){
     predicted_ratings <- edx_test_set %>%
         left_join(user_avgs, by='userId') %>%
         mutate(b_u = s/(n_i+l)) %>%
         mutate(pred = mu + b_u) %>%
         pull(pred)
     return(RMSE(predicted_ratings, edx_test_set$rating))
 })
lambda05 <- lambdas[which.min(rmses)]


lambda <- lambda05

user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda), n_i = n())
```

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

```r
predicted_ratings_algorithm_05 <- edx_test_set %>%
     left_join(user_reg_avgs, by='userId') %>%
     mutate(pred = mu + b_u) %>%
     pull(pred)

RMSE05 <- RMSE(predicted_ratings_algorithm_05, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9784941
```

Algorithm 6 (Average Rating plus Genre Effects). The sixth algorithm made predictions based on the EDX training set's average movie rating plus the effects of the genres of the movies being rated.

```r
mu <- mean(edx_train_set$rating)

genre_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = mean(rating - mu))
```

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

```r
predicted_ratings_algorithm_06 <- mu + edx_test_set %>%
     left_join(genre_avgs, by='genres') %>%
     pull(b_v)

RMSE06 <- RMSE(predicted_ratings_algorithm_06, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 1.017842
```

Algorithm 7 (Average Rating plus Regularized Genre Effects). The seventh algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the genres of the movies being rated.

```
lambdas <- seq(0, 30, 0.25)

mu <- mean(edx_train_set$rating)

genre_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(s = sum(rating - mu), n_i = n())
```

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

```
rmses <- sapply(lambdas, function(l){
     predicted_ratings <- edx_test_set %>%
         left_join(genre_avgs, by='genres') %>%
         mutate(b_v = s/(n_i+l)) %>%
         mutate(pred = mu + b_v) %>%
         pull(pred)
     return(RMSE(predicted_ratings, edx_test_set$rating))
 })
lambda07 <- lambdas[which.min(rmses)]

lambda <- lambda07

genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda), n_i = n())
```

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

```
predicted_ratings_algorithm_07 <- edx_test_set %>%
     left_join(genre_reg_avgs, by='genres') %>%
     mutate(pred = mu + b_v) %>%
     pull(pred)

RMSE07 <- RMSE(predicted_ratings_algorithm_07, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 1.017836
```

Algorithm 8 (Average Rating plus Month Effects). The eighth algorithm made predictions based on the EDX training set's average movie rating plus the effects of the months when the movies were rated.

```
mu <- mean(edx_train_set$rating)

monthly_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = mean(rating - mu))
```

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

```
predicted_ratings_algorithm_08 <- mu + edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(monthly_avgs, by='monthId') %>%
     pull(b_t)

RMSE08 <- RMSE(predicted_ratings_algorithm_08, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 1.05784
```

Algorithm 9 (Average Rating plus Regularized Month Effects). The ninth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the months when the movies were rated.

```
lambdas <- seq(0, 30, 0.25)

mu <- mean(edx_train_set$rating)

monthly_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(s = sum(rating - mu), n_i = n())
```

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

```
rmses <- sapply(lambdas, function(l){
     predicted_ratings <- edx_test_set %>%
         mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
         left_join(monthly_avgs, by='monthId') %>%
         mutate(b_t = s/(n_i+l)) %>%
         mutate(pred = mu + b_t) %>%
         pull(pred)
     return(RMSE(predicted_ratings, edx_test_set$rating))
 })
lambda09 <- lambdas[which.min(rmses)]

lambda <- lambda09

monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda), n_i = n())
```

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

```
predicted_ratings_algorithm_09 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_t) %>%
     pull(pred)

RMSE09 <- RMSE(predicted_ratings_algorithm_09, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 1.05784
```

Algorithm 10 (Average Rating plus Regularized Month and User Effects). The tenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the months when the movies were rated and the users registering the ratings.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
predicted_ratings_algorithm_10 <- edx_test_set %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(user_reg_avgs, by='userId') %>%
     mutate(pred = mu + b_i + b_u) %>%
     pull(pred)

RMSE10 <- RMSE(predicted_ratings_algorithm_10, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded a relatively low and promising RMSE of

```
## [1] 0.883397
```

Algorithm 11 (Average Rating plus Regularized Month and Genre Effects). The eleventh algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the months when the movies were rated and the genres of the movies being rated.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```
predicted_ratings_algorithm_11 <- edx_test_set %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     mutate(pred = mu + b_i + b_v) %>%
     pull(pred)

RMSE11 <- RMSE(predicted_ratings_algorithm_11, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9882456
```

Algorithm 12 (Average Rating plus Regularized Month and Movie Effects). The twelfth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the months when the movies were rated and the movies being rated.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```
predicted_ratings_algorithm_12 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_i + b_t) %>%
     pull(pred)

RMSE12 <- RMSE(predicted_ratings_algorithm_12, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9416049
```

Algorithm 13 (Average Rating plus Regularized User and Genre Effects). The thirteenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the users registering the ratings and the movies being rated.

```
mu <- mean(edx_train_set$rating)

user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```
predicted_ratings_algorithm_13 <- edx_test_set %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     mutate(pred = mu + b_u + b_v) %>%
     pull(pred)

RMSE13 <- RMSE(predicted_ratings_algorithm_13, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9431496
```

Algorithm 14 (Average Rating plus Regularized User and Month Effects). The fourteenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the users registering the ratings and the months when the ratings were registered.

```
mu <- mean(edx_train_set$rating)

user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```r
predicted_ratings_algorithm_14 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_u + b_t) %>%
     pull(pred)

RMSE14 <- RMSE(predicted_ratings_algorithm_14, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.980648
```

Algorithm 15 (Average Rating plus Regularized Movie, User, and Genre Effects). The fifteenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the movies being rated, the users registering the ratings, and the movies' genres.

```r
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```r
user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```r
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```r
predicted_ratings_algorithm_15 <- edx_test_set %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     mutate(pred = mu + b_u + b_v) %>%
     pull(pred)

RMSE15 <- RMSE(predicted_ratings_algorithm_15, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9431496
```

Algorithm 16 (Average Rating plus Regularized Movie, User, and Month Effects). The sixteenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the movies being rated, the users registering the ratings, and the months when the ratings were registered.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```
predicted_ratings_algorithm_16 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_i + b_u + b_t) %>%
     pull(pred)

RMSE16 <- RMSE(predicted_ratings_algorithm_16, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded another relatively low and promising RMSE, this time of

```
## [1] 0.8870891
```

Algorithm 17 (Average Rating plus Regularized Movie, Genre, and Month Effects). The seventeenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the movies being rated, the movies' genres, and the months when the ratings were registered.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```
predicted_ratings_algorithm_17 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_i + b_v + b_t) %>%
     pull(pred)

RMSE17 <- RMSE(predicted_ratings_algorithm_17, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9869919
```

Algorithm 18 (Average Rating plus Regularized User, Genre, and Month Effects). The eighteenth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the users reigsetring the ratings, the movies' genres, and the months when the ratings were registered.

```
mu <- mean(edx_train_set$rating)

user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```
predicted_ratings_algorithm_18 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_u + b_v + b_t) %>%
     pull(pred)

RMSE18 <- RMSE(predicted_ratings_algorithm_18, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9457482
```

Algorithm 19 (Average Rating plus Regularized Movie, User, Genre, and Month Effects). The nineteeth algorithm made predictions based on the EDX training set's average movie rating plus the regularized effects of the movies being rated, the users submitting the ratings, the movies's genres, and the months when the ratings were registered.

```
mu <- mean(edx_train_set$rating)

movie_reg_avgs <- edx_train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda03), n_i = n())
```

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

```
user_reg_avgs <- edx_train_set %>%
     group_by(userId) %>%
     summarize(b_u = sum(rating - mu)/(n()+lambda05), n_i = n())
```

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

```
genre_reg_avgs <- edx_train_set %>%
     group_by(genres) %>%
     summarize(b_v = sum(rating - mu)/(n()+lambda07), n_i = n())
```

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

```
monthly_reg_avgs <- edx_train_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     group_by(monthId) %>%
     summarize(b_t = sum(rating - mu)/(n()+lambda09), n_i = n())
```

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

```
predicted_ratings_algorithm_19 <- edx_test_set %>%
     mutate(monthId = round_date(as_datetime(timestamp), unit = "month")) %>%
     left_join(movie_reg_avgs, by='movieId') %>%
     left_join(user_reg_avgs, by='userId') %>%
     left_join(genre_reg_avgs, by='genres') %>%
     left_join(monthly_reg_avgs, by='monthId') %>%
     mutate(pred = mu + b_i + b_u + b_v + b_t) %>%
     pull(pred)

RMSE19 <- RMSE(predicted_ratings_algorithm_19, edx_test_set$rating)
```

When applied to the EDX test set, this algorithm yielded an RMSE of

```
## [1] 0.9460756
```

Results

The following table summarizes the results achieved via the nineteen algorithms:

```
##       Algorithm      RMSE
## 1  Algorithm 10 0.8833970
## 2  Algorithm 16 0.8870891
## 3  Algorithm 12 0.9416049
## 4  Algorithm 13 0.9431496
## 5  Algorithm 15 0.9431496
## 6  Algorithm 03 0.9432842
## 7  Algorithm 02 0.9433553
## 8  Algorithm 18 0.9457482
## 9  Algorithm 19 0.9460756
## 10 Algorithm 05 0.9784941
## 11 Algorithm 04 0.9791190
## 12 Algorithm 14 0.9806480
## 13 Algorithm 17 0.9869919
## 14 Algorithm 11 0.9882456
## 15 Algorithm 07 1.0178360
## 16 Algorithm 06 1.0178421
## 17 Algorithm 09 1.0578403
## 18 Algorithm 08 1.0578404
## 19 Algorithm 01 1.0604284
```

The best algorithm was

```
## [1] 10
```

Conclusion

Summary: The purpose of this project was to select an algorithm capable of predicting movie ratings with a commercially viable RMSE. Nineteen algorithms were developed and evaluated. The one selected at the end of the evaluation process accounted for the average rating in the EDX training set as well as two other predictors: (a) the movie being rated and (b) the user registering the rating. It generated an RMSE of less than 0.884 against the EDX test set and should generate a similar RMSE against the VALIDATION set.

Limitations: A review of literature regarding the NETFLIX challenge suggests lower RMSEs could be achieved by incorporating the matrix factorization concepts outlined in Section (33.11.12) of Professor Irizarry's "Introduction to Data Science." The same section offers that the recommenderlab package, outside the scope of the course, is essential to the application of those concepts. Among those project's limitations were ones stemming from the non-use of recommenderlab.

Future Work: The author is interested in exploring recommenderlab and learning more about the advanced application of matrix factorization concepts. That future work may generate a lower and, therefore, more commercially viable RMSE.