Movie Recommendation System

KH W

02/08/2021

Overview

In this project, a system is developed to predict users' rating on movies they have not watched before. The system is being trained by a dataset and it is obtained at http://files.grouplens.org/datasets/movielens/ml-10m.zip. The dataset consists of 6 columns: User ID, Movie ID, Movie Title, Genre, Rating, and Timestamp of the rating. 90% of this dataset (9,000,055 rows) is defined as the training dataset and is used for developing the system. The remaining 10% (999,999 rows) is defined as test dataset which is treated as new data for the final model. The prediction will be compared to the actual ratings in the test dataset and the root mean squared error (RMSE) will be reported.

The methodology is to mine appropriate and useful data structures from the training dataset, transform these structures into predictors of the model. The model is a linear regression with the best model selected by forward selection.

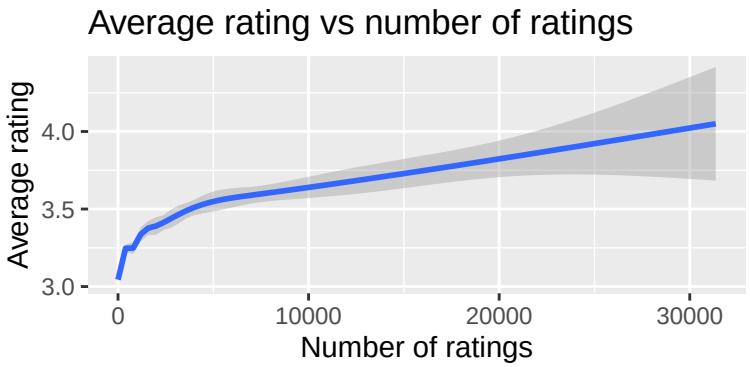
The goal of this project is to recommend movies to the users, therefore the ultimate product is a list of recommendation with the most recommended movie being listed on the top of the list and followed by second recommended, and so on.

Dataset exploration

Movie popularity

The rating data suggested the more rating a movie has, the higher rating that movie has. This could also be explained by intuition that highly rated movies are popular and popular movies have good ratings. A plot would be useful to illustrate the pattern:

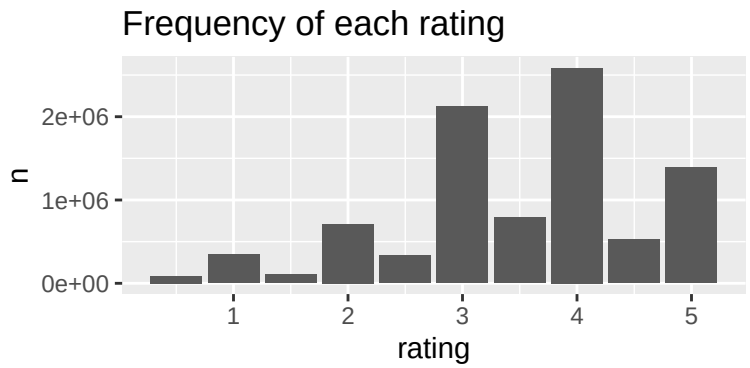
```
plot_moviepop
```



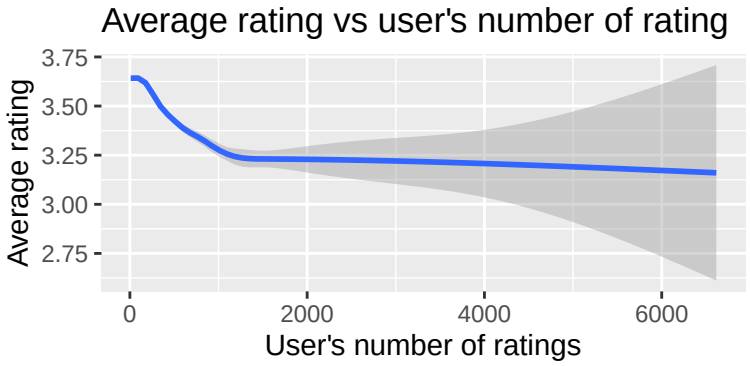
Frequent raters

The data suggests frequent raters tend to rate lower than non-frequent raters. This could be explained because frequent raters watch more movies means they are more likely to experience low quality movies (i.e., rating less than or equal to 3) which are comparatively uncommon in the movie database (see the rating distribution below). A graph could show the relationship between users' rating frequency and their average ratings:

```
plot_rating_hist
```



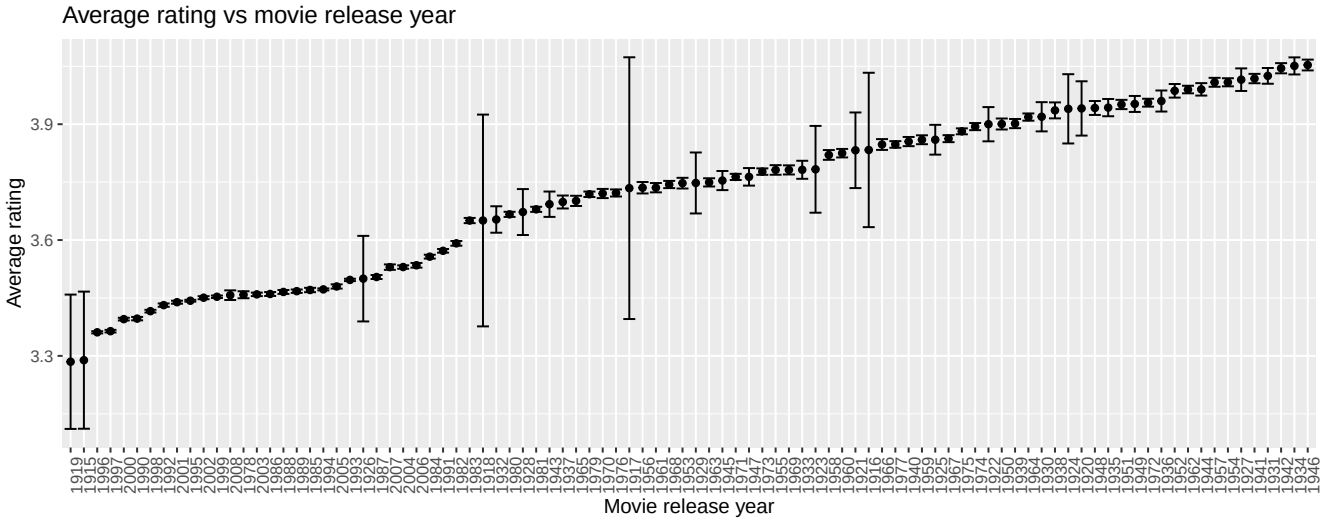

```
plot_freqrater
```



Movie released year

Released year of movies could be informative to the model because most of the 95% confidence mean rating of each released year are narrow and not overlapping with each other. The plot shows the 95% confidence interval of each year sorting from the worst rated year to the best rated year:

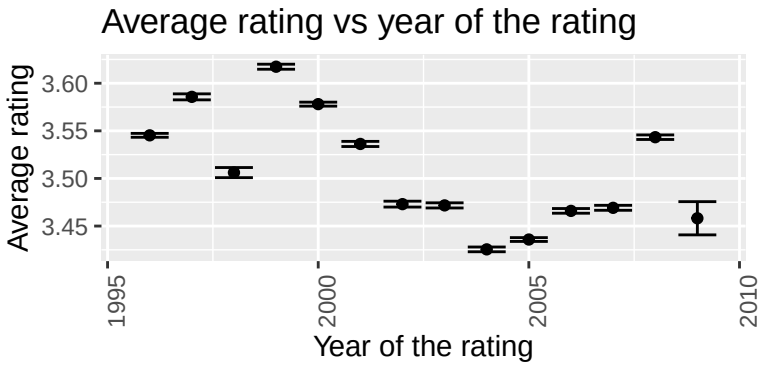
```
plot_movieyear
```



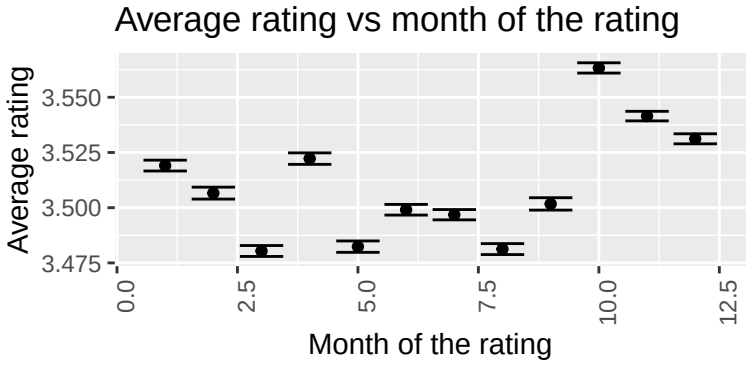
Rating timestamp

Similarly, the timestamp of the rating could be informative. The confidence intervals do not overlap with each other.

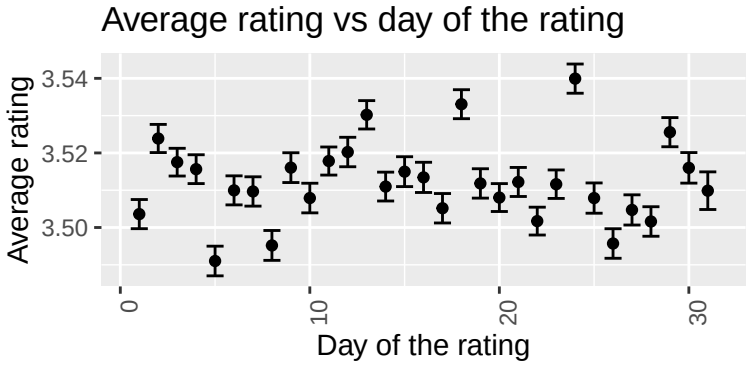
```
plot_rateyear
```



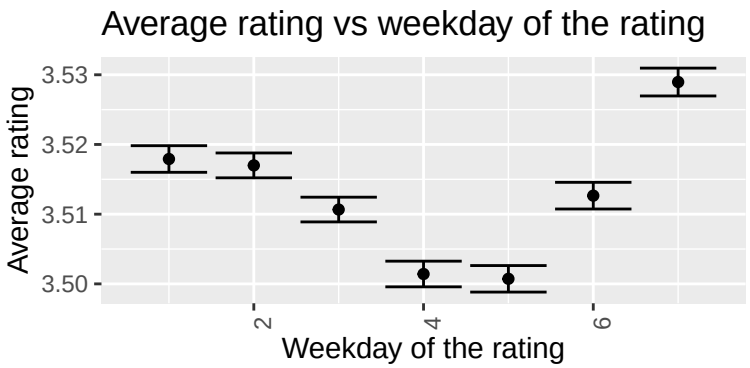

```
plot_ratemonth
```



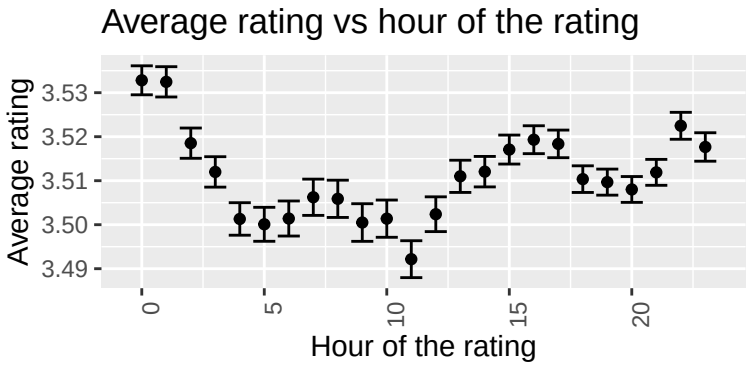

```
plot_rateday
```




```
plot_rateWday
```



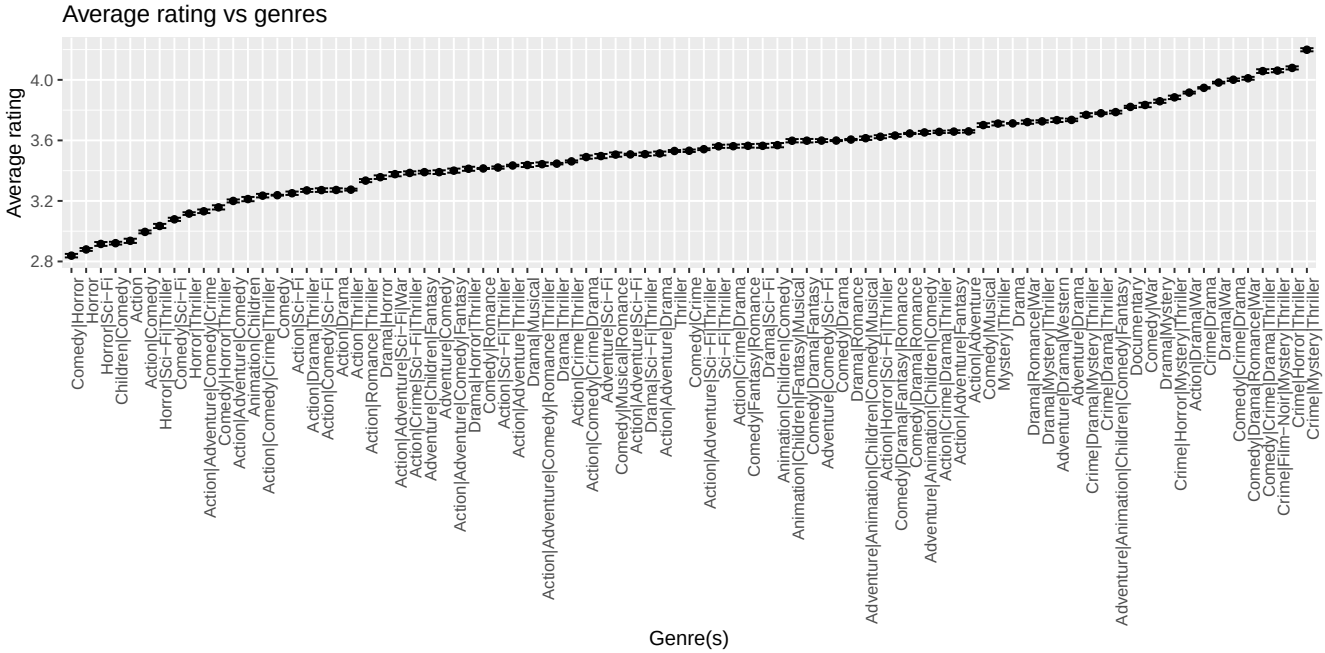

```
plot_ratehour
```



Genres

Genres of the movies, as expected, have impact to the predictive power of the model. By sorting out the most frequently rated genres and plot it against the average rating, one can see the confidence intervals do not overlap with each other and the average rating ranges from 2.8 to more than 4.

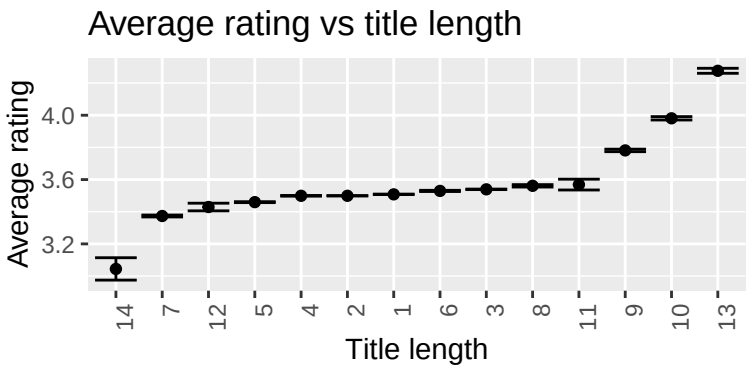
```
plot_genre
```



Movie title length

Surprisingly, the length of the movie title also suggest a rating pattern. Titles with 14 words has a mean rating of below 3.2, titles with 13 words has a mean rating of above 4.

```
plot_titlelen
```



All the sections above will be considered as predictors of the model.

Model performance

The model is a linear regression. The best model is selected by forward selection. The forward selection process starts with 3 predictors (mean of each movie, mean of each user, movie popularity) in the model, then the process adds the most contributive predictor until the improvement is no longer significant.

The final model is:

```
summary(lmfitbest)
```

```
##
## Call:
## lm(formula = rating ~ movieMean + userMean + n_movie + n_user +
##     rateYear + movieYear + movieMeanStdError + userMeanStdError +
##     rateMonth + userSD + movieSD + genreMeanStdError + genreMean +
##     titlewords + rateWday + rateDay + rateHour, data = edx[,
##     -c("userId", "movieId", "timestamp", "title", "genres")])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6669 -0.4987  0.0674  0.5839  5.0525
##
## Coefficients:
##                     Estimate Std. Error  t value Pr(>|t|)
## (Intercept)        1.780e+01  1.719e-01  103.516  < 2e-16 ***
## movieMean          9.204e-01  8.944e-04 1029.094  < 2e-16 ***
## userMean           8.703e-01  7.756e-04 1122.103  < 2e-16 ***
## n_movie           -1.979e-06  5.144e-08  -38.482  < 2e-16 ***
## n_user             6.672e-05  6.903e-07   96.648  < 2e-16 ***
## rateYear          -1.149e-02  8.630e-05 -133.141  < 2e-16 ***
## movieYear          1.274e-03  2.329e-05   54.726  < 2e-16 ***
## movieMeanStdError  3.791e-01  1.044e-02   36.304  < 2e-16 ***
## userMeanStdError  -3.151e-01  8.235e-03  -38.263  < 2e-16 ***
## rateMonth         -1.846e-03  8.340e-05  -22.136  < 2e-16 ***
## userSD             3.759e-02  1.688e-03   22.269  < 2e-16 ***
## movieSD           -6.060e-02  3.547e-03  -17.085  < 2e-16 ***
## genreMeanStdError  7.046e-03  4.697e-04   15.003  < 2e-16 ***
```

```
## genreMean         -1.922e-02  1.503e-03  -12.790  < 2e-16 ***
## titlewords         1.177e-03  1.661e-04    7.083 1.41e-12 ***
## rateWday          -8.053e-04  1.486e-04   -5.419 6.00e-08 ***
## rateDay           -1.790e-04  3.298e-05   -5.427 5.73e-08 ***
## rateHour          -8.983e-05  3.975e-05   -2.260   0.0238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8701 on 9000037 degrees of freedom
## Multiple R-squared:  0.3267, Adjusted R-squared:  0.3267
## F-statistic: 2.569e+05 on 17 and 9000037 DF,  p-value: < 2.2e-16
```

The RMSE of the model on the training dataset is 0.87002 and the RMSE of the model on the validation dataset is 0.87755.

```
RMSE(edx$y_hat,edx$rating)
```

```
## [1] 0.8700243
```

```
RMSE(validation$y_hat,validation$rating)
```

```
## [1] 0.8775491
```

Conclusion

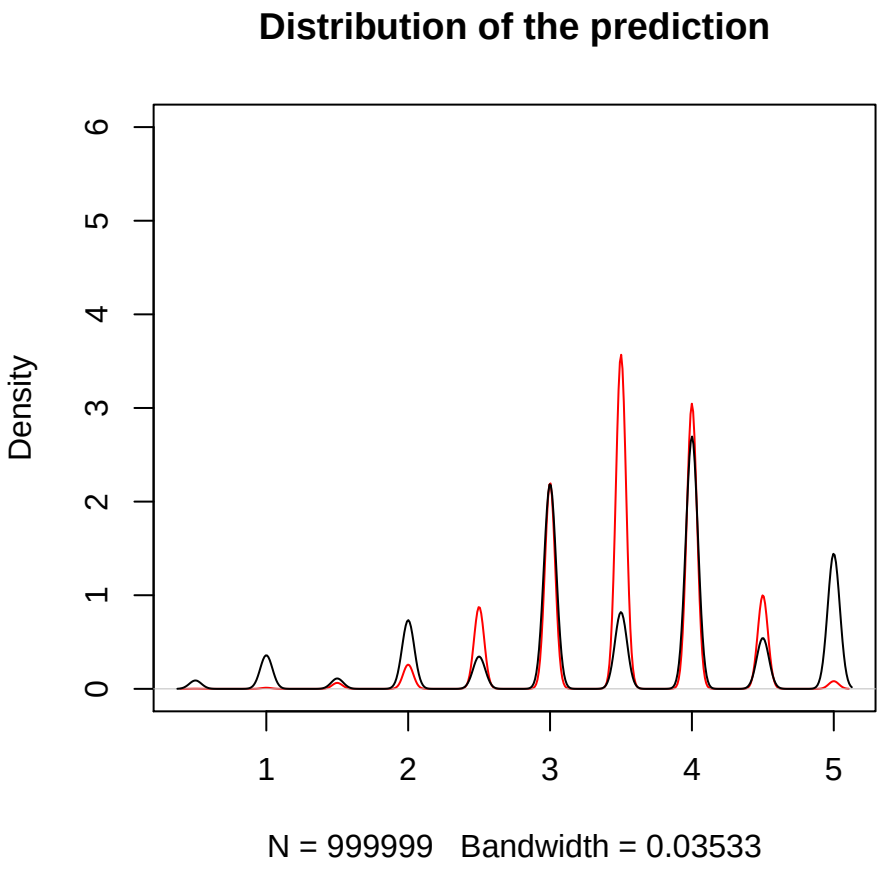
The system can recommend movies to any user starting from the highest estimated rating. For example, for user 31063, the list of movie recommendation sorting from most recommended to least recommended, is:

```
recommendations %>% filter(userId==31063)
```

```
## # A tibble: 11 x 4
## # Groups:   userId [1]
##    userId movieId title                                          y_hat
##     <int>   <dbl> <chr>                                          <dbl>
##  1  31063    1199 Brazil (1985)                                   3.65
##  2  31063    1258 Shining, The (1980)                             3.61
##  3  31063    1266 Unforgiven (1992)                               3.60
##  4  31063     150 Apollo 13 (1995)                                3.48
##  5  31063    1307 When Harry Met Sally... (1989)                  3.47
##  6  31063    1017 Swiss Family Robinson (1960)                    3.14
##  7  31063     592 Batman (1989)                                   3.02
##  8  31063      81 Things to Do in Denver When You're Dead (1995)  2.99
##  9  31063      95 Broken Arrow (1996)                             2.80
## 10  31063     382 Wolf (1994)                                     2.75
## 11  31063    1391 Mars Attacks! (1996)                           2.65
```

Using this system, users can watch movies they probably like without looking up by themselves. One of the limitations of this system is it rarely predicts extreme rating (i.e. 5 stars or 0.5 stars). By rounding the prediction to the nearest 0.5 (in red), and compare to the actual ratings, one can see the system rarely predicts 0.5 to 2 stars and 5 stars:

```
plot(density(round(validation$y_hat*2,0)/2),
     main = "Distribution of the prediction",
     ylim = c(0,6), col = "red")
lines(density(round(validation$rating*2,0)/2))
```



This means the prediction cannot identify user's clear preferences, such as favorite actors, favorite directors or movies in sequels (e.g. Star Wars, Matrix). This limitation can be eliminated if there are extra attributes of the movies.