HarvardX PH125.9x Data Science: MovieLens Project Submission

Kai Yin Phyllis Lun

3/7/2022

Data Cleaning

Edx set and validation set

Here I first separated the movie titles from the year of movie release. I have also converted the time stamps of review to standard date formate and extracted the years of rating. A new variable, year_diff, is used to assess the time difference between year of movie release and year of rating.

```
edx_clean <- edx%>%
  mutate(year=str_extract(title, "\\(\\d+\\)")) %>%
  mutate(year=str_extract(year, "\\d+")) %>%
  mutate(title=str_remove(title, "\\s+\\(\\d+\\)")) %>%
  mutate(rate_year= as.POSIXct(timestamp,origin = "1970-01-01"))%>%
  mutate(rate_year=as.numeric(format(rate_year,"%Y")),
         year=as.numeric(str_extract(year,"\\d+"))) %>%
  select(-timestamp) %>% mutate(year_diff=rate_year-year)
```

The title of movie ID #889 has to be manually processed due to it having two brackets of years in the title.

```
edx_clean[edx_clean$movieId==889]<-edx_clean[edx_clean$movieId==889] %>%
  mutate (year=as.numeric(1994))   %>% mutate(year_diff=rate_year-year)
```

The same process is applied to the validation set so that the machine learning models can be applied properly later on.

```
validation_clean<-validation%>%
  mutate(year=str_extract(title, "\\(\\d+\\)")) %>%
  mutate(year=str_extract(year, "\\d+")) %>%
  mutate(title=str_remove(title, "\\s+\\(\\d+\\)")) %>%
  mutate(rate_year= as.POSIXct(timestamp,origin = "1970-01-01"))%>%
  mutate(rate_year=as.numeric(format(rate_year,"%Y")),
         year=as.numeric(str_extract(year,"\\d+"))) %>%
  select(-timestamp) %>% mutate(year_diff=rate_year-year)

validation_clean[validation$movieId==889]<-
  validation_clean[validation_clean$movieId==889] %>%
  mutate (year=as.numeric(1994))  %>% mutate(year_diff=rate_year-year)
```

Data Exploration

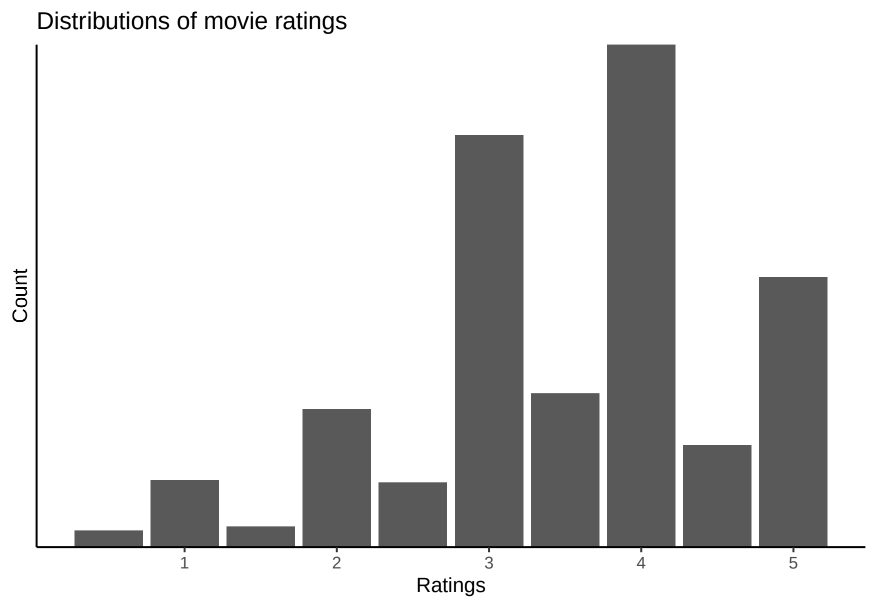
In the edx dataset, we have 69878 unique users and 10677 unique movies.

```
edx_clean %>% summarize(n_users = n_distinct(userId),
                        n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

I. Distribution of movie ratings

The most frequently given ratings are 3 and 4.



II. Distribution of number of ratings by users

We can see that there is a huge variation on the number of movie ratings given by the users (range=10-6616). Three quarters of the users rating 141 movies or less. Of the 69878 users included in the pilot set, on average they have 129 move ratings and three quarters of them have about 141 ratings. That means a quarter of users gave a large number of movie ratings.

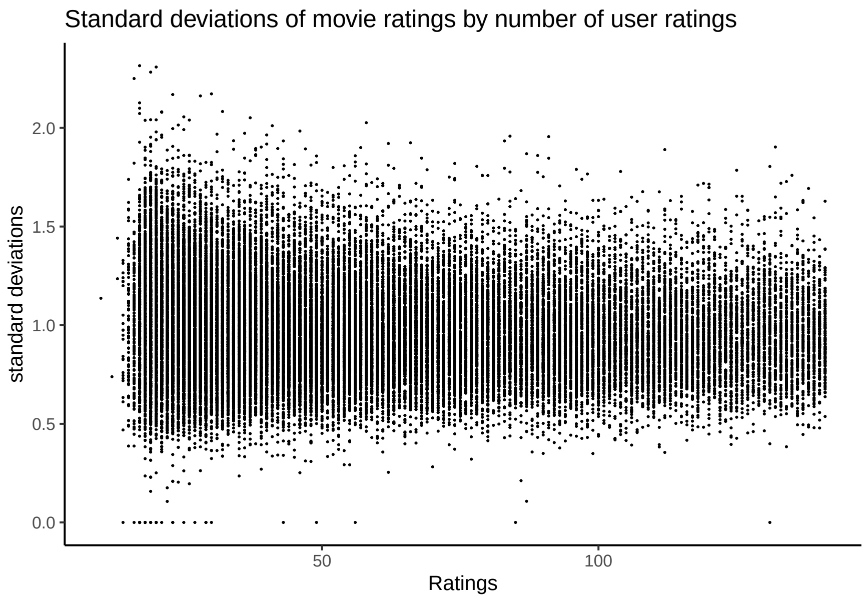
```
edx_clean %>% group_by(userId) %>% count(userId)  %>% summary(n)
```

```
##      userId            n
##  Min.   :    1   Min.   :  10.0
##  1st Qu.:17943   1st Qu.:  32.0
##  Median :35798   Median :  62.0
##  Mean   :35782   Mean   : 128.8
##  3rd Qu.:53620   3rd Qu.: 141.0
##  Max.   :71567   Max.   :6616.0
```

III. Association between the variations of movie ratings and prolific raters

Generally, ratings coming from prolific raters are more similar than those from users who provided only a few ratings.However, it should be noted that the deviation is quite large among those with less than or equal to 141 movie ratings.

```
edx_clean %>% group_by(userId) %>% summarize(standard_dev=sd(rating)) %>%
  left_join(edx_clean %>% group_by(userId) %>% count(userId))%>%
  filter(n<=141) %>%
  as_tibble() %>%
  ggplot(aes(x=n,y=standard_dev)) + geom_point(size=0.1) +
  scale_x_continuous()+
  labs(title="Standard deviations of movie ratings by number of user ratings",
       x="Ratings",
       y="standard deviations", face="bold")+
  theme_classic()
```



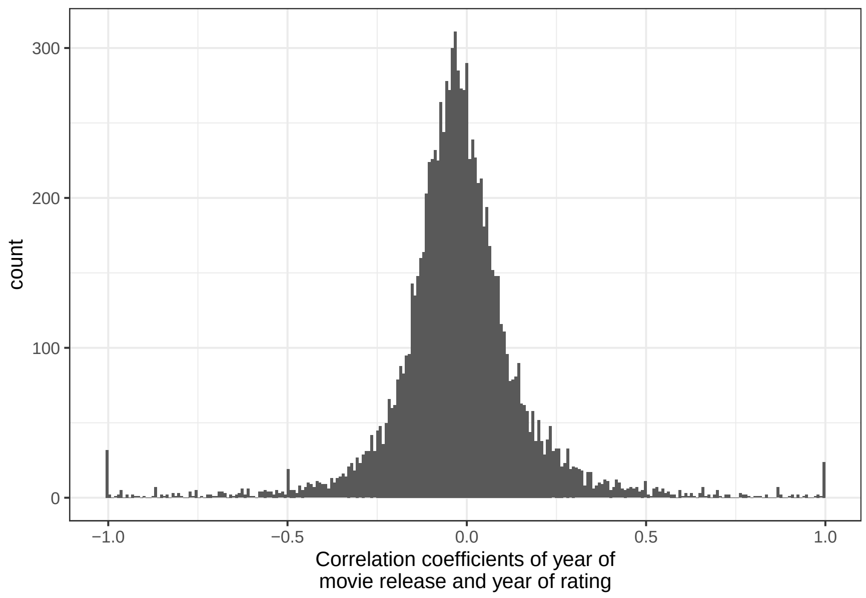
IV. Number of ratings received by a movie ——— Similar to the number of ratings by users, the number of ratings received by movies also varied greatly (range = 1-31362). Three quarters of movies received less than 565 ratings.

```
edx_clean %>% dplyr::count(movieId) %>% summary()
```

```
##     movieId            n
##  Min.   :    1   Min.   :    1.0
##  1st Qu.: 2754   1st Qu.:   30.0
##  Median : 5434   Median :  122.0
##  Mean   :13105   Mean   :  842.9
##  3rd Qu.: 8710   3rd Qu.:  565.0
##  Max.   :65133   Max.   :31362.0
```

V. Year of movie release and year of movie rating

The correlation between the year of rating and the year of movie release appears roughly normally distributed.



Model Training

Evaluation metric: RMSE

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

Setting up training and testing sets within the edx dataset

Within the edx dataset, I am creating a training set and a test set. The training set constitutes 90% of the edx dataset. There are 8100048 observations in the training set.

```
## [1] 8100048
```

Model 1: Movie effect model

The movie effect model has a RMSE of 0.9442066.

```
## [1] 16
```

method	RMSE
edx: movidId only model	0.9442066

Model 2: Movie and user effect model

The movie and user effect model has a RMSE of 0.8663173, which is lower than the movie effect only model.

method	RMSE
edx: movidId only model	0.9442066
edx: movidId and userId model	0.8663173

Model 3: Movie, user, and difference-in-year effect model

The Movie, user, and difference-in-year effect model has the lowest RMSE thus far: 0.8659055.

method	RMSE
edx: movidId only model	0.9442066
edx: movidId and userId model	0.8663173
edx: movidId, userId, and difference-in-year model	0.8659055

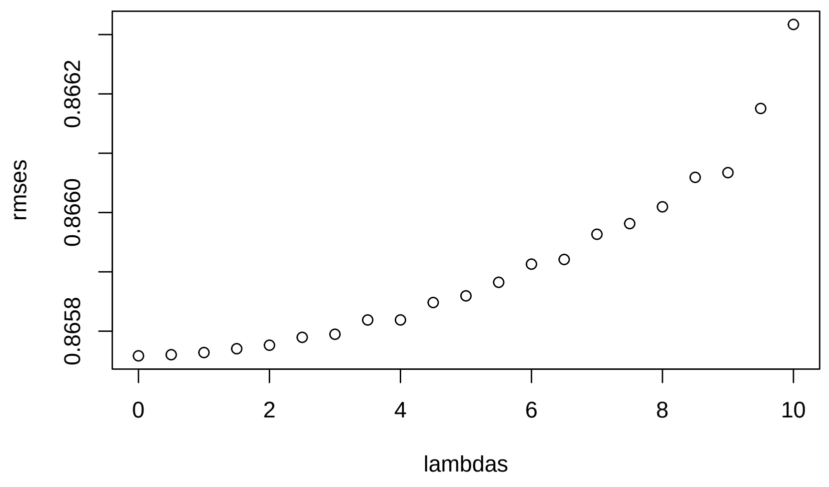
Model 4: Movie, user, difference-in-year, and movie genre effect model

Lastly, the movie, user, difference-in-year, and movie genre effect model only improved the last model slightly, with a RMSE of 0.8655948.

method	RMSE
edx: movidId only model	0.9442066
edx: movidId and userId model	0.8663173
edx: movidId, userId, and difference-in-year model	0.8659055
edx: movidId, userId, difference in year, and genre model	0.8655948

Testing regularization in Model 4: Movie, user, difference-in-year, and movie genre effect model

It turns out lamda=5 in the regularization model would achieve the lowest RMSE.



```
## [1] 5
```

Applying trained, regularized machine learning models to validation dataset.

Regularized Model 4: Movie, user, difference-in-year, and movie genre effect model

Finally, the fourth and final model has a RMSE of 0.8652284.

method	RMSE
edx: movidId only model	0.9442066
edx: movidId and userId model	0.8663173
edx: movidId, userId, and difference-in-year model	0.8659055
edx: movidId, userId, difference in year, and genre model	0.8655948
validation: movidId, userId, difference in year, and genre model (regularized)	0.8652284

Conclusion

After fitting the four trained linear regression models to the validation data and in consideration of the constraints of these models, the model with the best performance is the regularized model with movidId, userId, difference in years of movie release and movie rating ; and movie genres as features (lambda = 5), which yields a RMSE of 0.8652284.