MovieLens Capstone Project

Phil Girurugwiro

2/15/2020

MovieLens Project

Project Introduction, Overview, and Executive Summary

This report documents the analysis that was completed on the Movielens data set of 10 million movie ratings. The objective was to build a model that would predict the ratings of a given subset of the movies with a relatively good accuracy, i.e. with minimum loss function (RMSE). To accomplish this task, the data set was divided into 3 distincts data sets. First, the data was divided into a working set referred to as the "edx" set in the project and a "validation" set for which the final ratings were to be predicted and the final RMSE value to be determined. The edx set was subsequently divided into two sets, a training set and a test set. The training set was used to train the model and the test set was used to test the accuracy of the model. Once a good model was achieved, it was then used to make the predictions on the validation set.

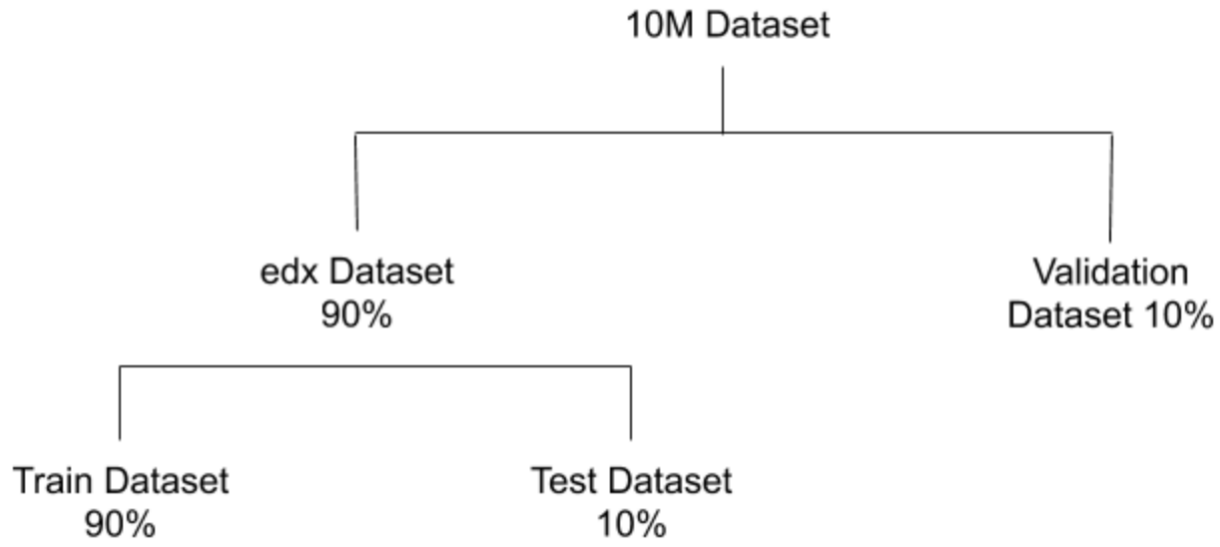

Figure 1: Fig 1. Data Partition

The analysis involved studying the data set and gaining insight on the trends and patterns of each feature of the data through exploratory data analysis. Different models using different features (predictors) and combinations of features were built and tested. The models were compared against each other and the final, best, model was used on the validation set and used to calculate the final RMSE. The final RMSE was determined to be 0.8647.

Analysis and Methods

1. Exploratory Data Analysis

The data set was downloaded from the provided link and processed to extract useful data that would later be used in the training and validation of the model. This step included extracting the movie release year from the movie title, and changing the timestamp of the rating from computer time format to actual date format.

```
movielens <- movielens %>% mutate(rating_date = as_datetime(timestamp), rating_year = as.character(forma
  mutate(release_year = str_extract(movielens$title,  "\\d{4}"))
```

The following items were sought to be understood:

- Understand the distribution of users vs. number of ratings per user: This steps helps visualize the variation in activity among users, i.e. do all users rate the same amount of movies or are some more active than others?
- Understand the distribution of movies vs. number of ratings per movie. This steps helps visualize if some movies get more ratings than others.
- Understand the distribution of the number of ratings vs. the release year of the movie. Understand if certain release years have more ratings than others
- Understand the average rating of the movies vs. release year of the movie. Did some years have movies with better ratings than others?
- Understand the change of the average rating of the movies overtime. Are movies getting higher or lower ratings as time goes by since their release?
- Understand the effect of movie genre to the rating.

Distribution of Users vs. Number of Ratings

```
edx %>% group_by(userId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30) +
  scale_x_log10() +
  xlab("Total Ratings per User") +
  ylab("Number of Users")
```

The majority of users have rated between 50 and 200 movies. A small number of very active users have rated more than 1000 movies. It can be concluded that there's enough data per user to make adequate predictions.

Distribution of Movies vs. Number of Ratings

```
edx %>% group_by(movieId) %>%
  summarize(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30) +
  scale_x_log10() +
  xlab("Total Ratings per Movie") +
  ylab("Number of Movies")
```

The majority of movies have between 50 and 1000 ratings. This distribution follows a quasi-normal distribution centered around 100 ratings per movie. There's enough rating data per movie to study the impact of individual movie ratings.

Number of Ratings vs. Release Year

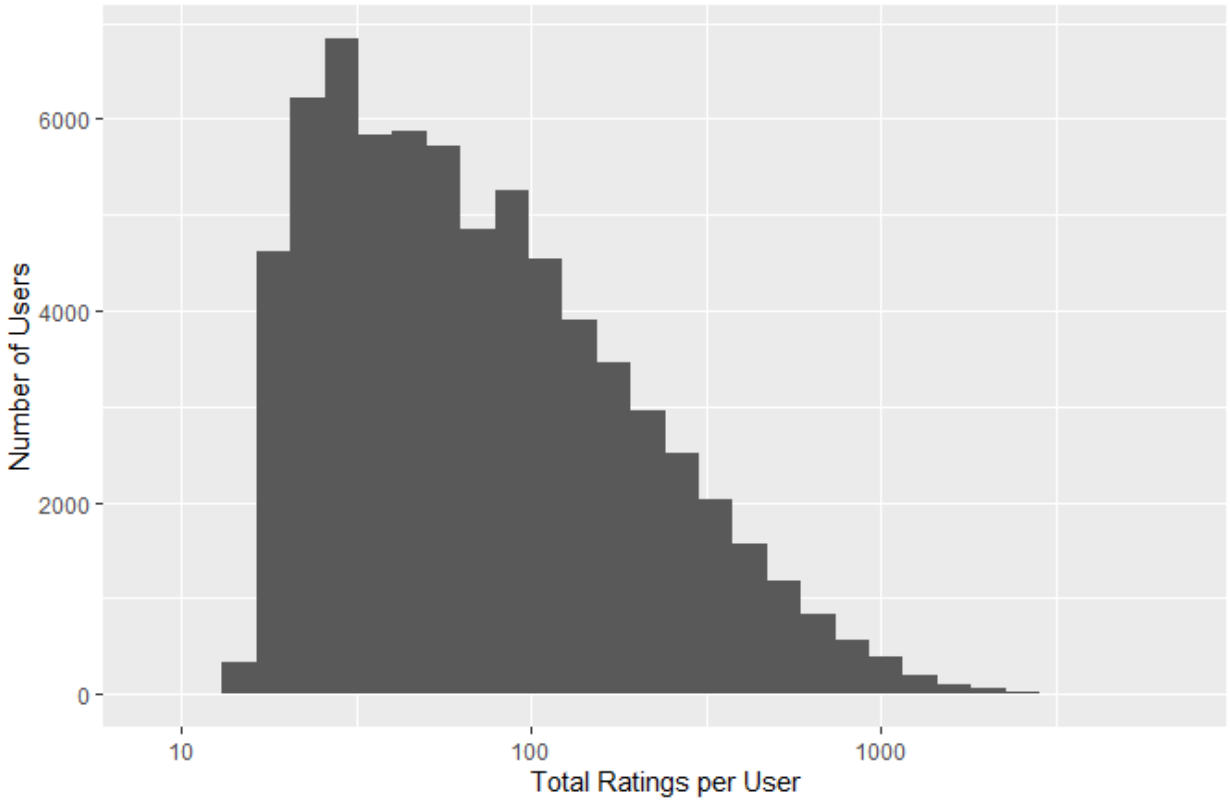

Figure 2: Fig 2. Users vs. Number of Ratings per User

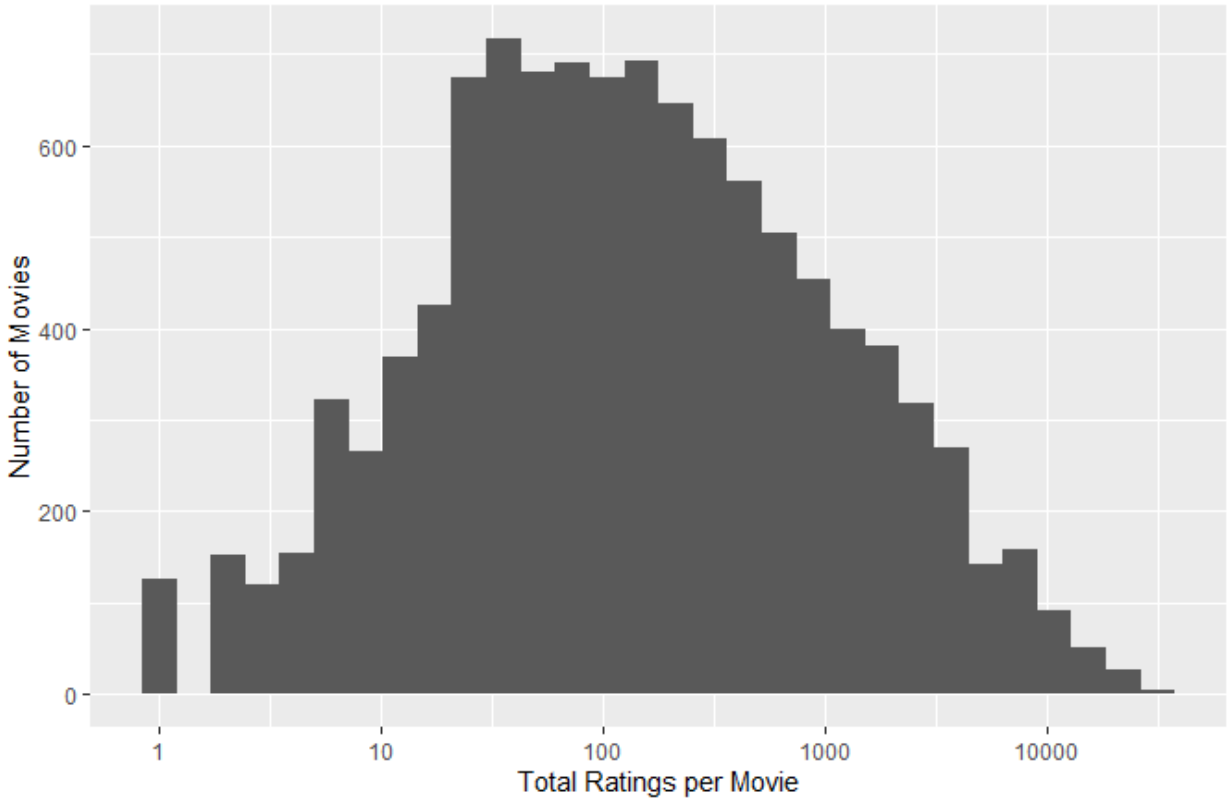

Figure 3: Fig 3. Movies vs. Number of Ratings per Movie

```
edx %>% group_by(movieId) %>%
  summarize(N_ratings = n(), year_out = as.character(first(release_year))) %>%
  ggplot(aes(year_out, N_ratings)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_y_log10()+
  xlab("Release Year")+
  ylab("Total Ratings")
```

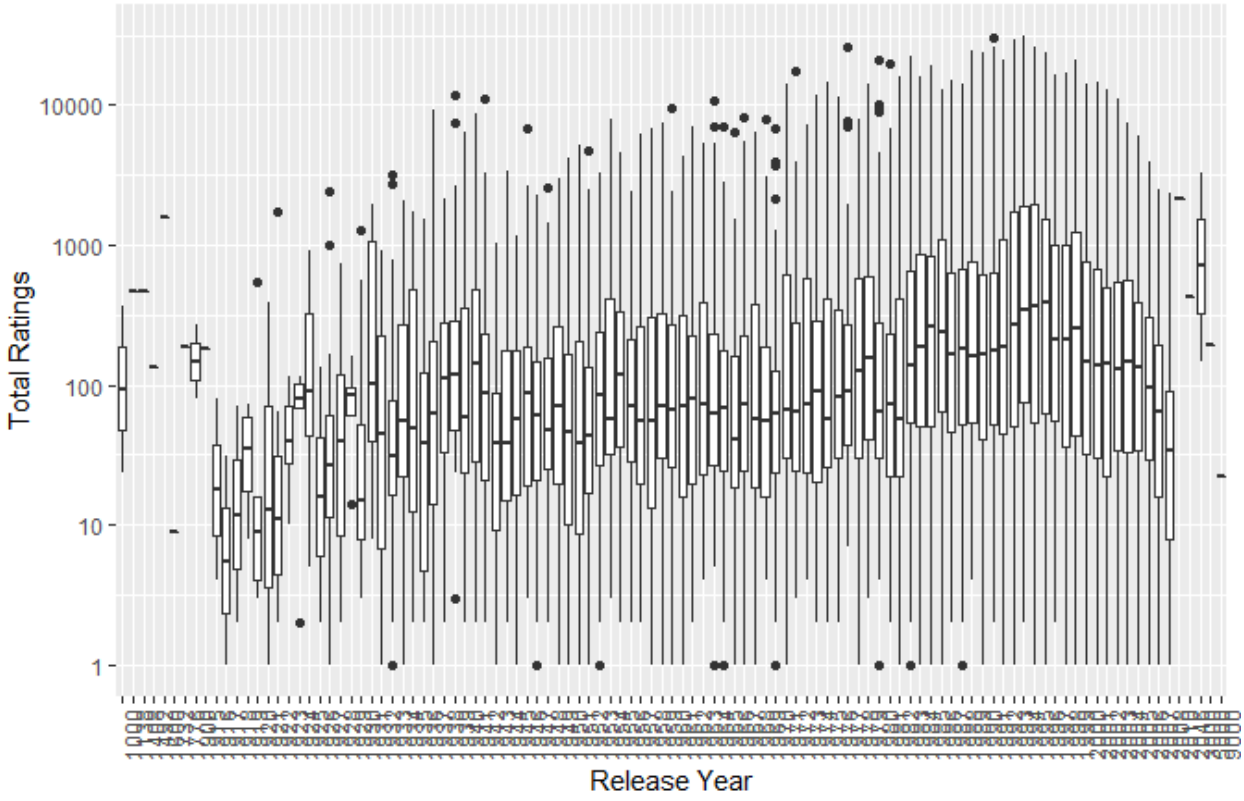

Figure 4: Fig 4. Number of Ratings vs. Release Year

There is a visible uptrend in the median number of ratings per movie release year peaking in 1995 but hovers around the mean of 100 ratings. This effect will not be considered in model training.

Average rating of the movie vs. Release Year

```
edx %>% group_by(movieId) %>%
  summarize(N_ratings = n(), year_out = as.character(first(release_year))) %>%
  ggplot(aes(year_out, N_ratings)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_y_log10()+
  xlab("Release Year")+
  ylab("Total Ratings")
```

There is a visible downward trend that can be explain by the fact that as more movies came out, a wider mix of good movies and bad movies came out driving down the overall average. This effect must be accounted for in the training of the model by regularizing respective parameters.

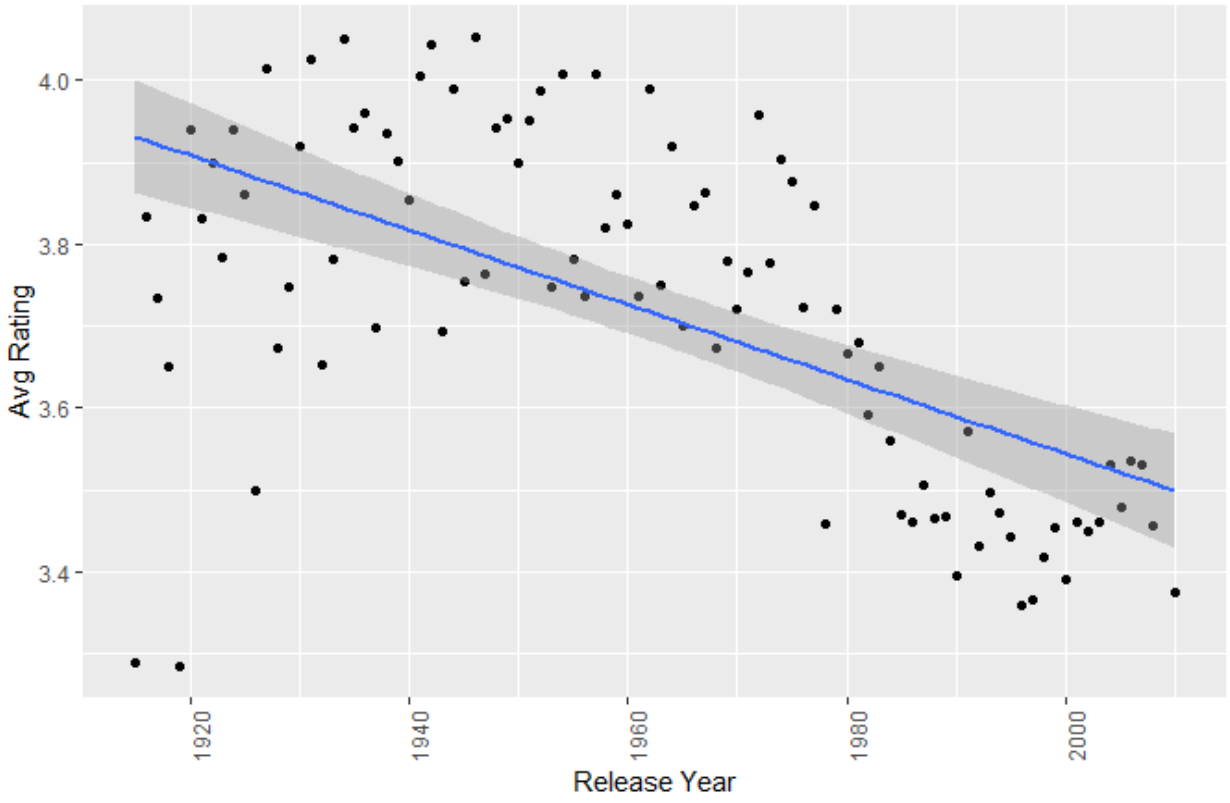

Figure 5: Fig 5. Avg Rating vs. Release Year

Average rating of the movie vs. time of rating

```
edx %>% mutate(rating_date = round_date(rating_date, unit = "week")) %>%
  group_by(rating_date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(rating_date, rating)) +
  geom_point() +
  geom_smooth(method = "lm") +
  xlab("Time of Rating")+
  ylab("Avg Rating")
```

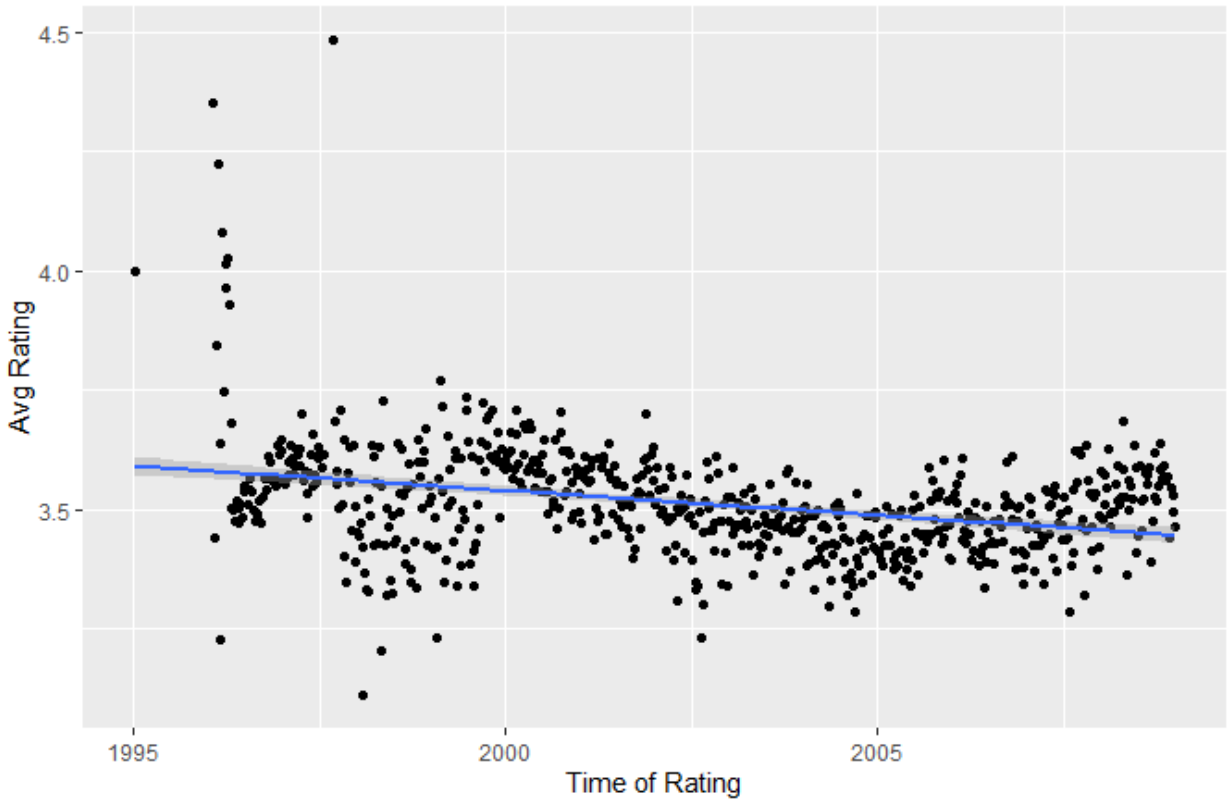

Figure 6: Fig 6. Avg Rating vs. Rating Date

There is a visible downward trend, which indicates that movies tend to get a better rating when they first come out but this effect attenuates overtime as users get used to the movies. Time of rating effects must be accounted for in the prediction model.

Effect of genre

```
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 50000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
```

```
  xlab("Movie Category (genres)") +
  ylab("Average Rating")
```

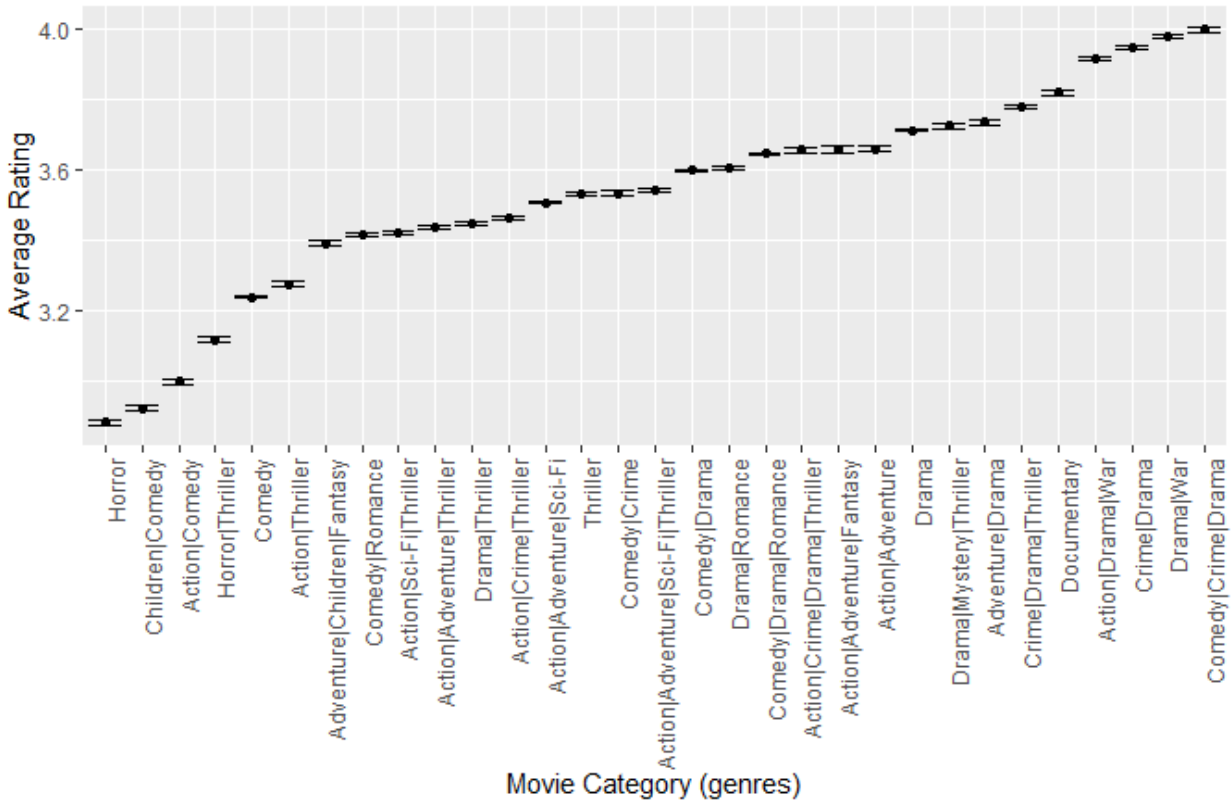

Figure 7: Fig 7. Avg Rating vs. Genre Categories

It is clear that some genres get better ratings than others.

2. Model Training and Evaluation

Training the model consisted of several steps. On each step, the training set was used to train the model and the tes set was used to evaluate the model by calculating the resulting RMSE.

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

The final model was used on the Validation set.

The steps are as follows:

- Predict using just the average rating

```
mu <- mean(train_set$rating)
predicted_ratings_0 <- rep(mu, length(test_set$rating))
model_0_rmse <- RMSE(predicted_ratings_0, test_set$rating)
model_0_rmse
```

Giving each movie in the test set a rating that is equal to the overall average rating in the train set results in an RMSE of 1.06. Let's consider the effect of other parameters discussed in the exploratory data analysis.

- Predict with movie effect considerations

```
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings_1 <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>% .$b_i
model_1_rmse <- RMSE(predicted_ratings_1, test_set$rating)
model_1_rmse
```

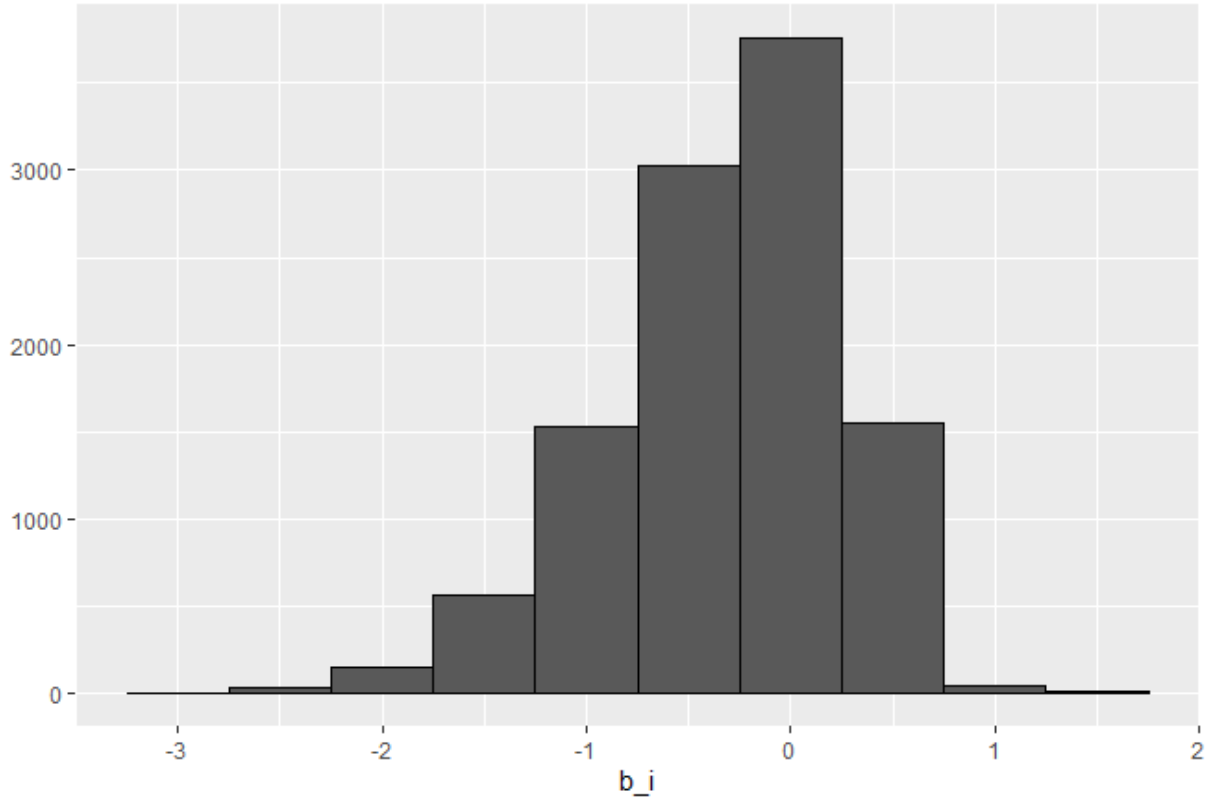

Figure 8: Fig 8. Movie Effects, b_i

Adding movie effects to the model improves the RMSE to 0.943.

- Predict with user effect considerations

```
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i)
predicted_ratings_2 <- test_set %>%
  left_join(user_avgs, by = "userId") %>%
  left_join(movie_avgs, by = "movieId") %>%
```

```
  mutate(prediction =mu+b_i+b_u) %>%
  pull(prediction)
model_2_rmse <- RMSE(predicted_ratings_2, test_set$rating)
model_2_rmse
```

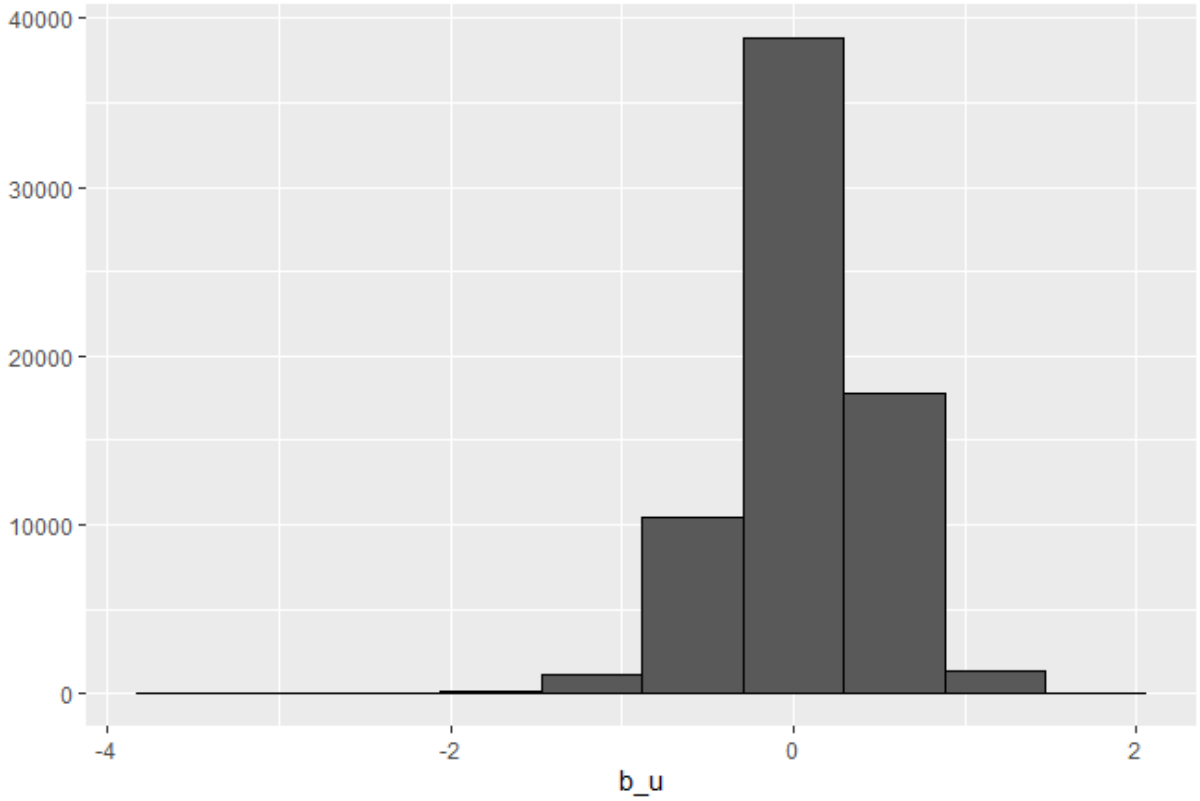

Figure 9: Fig 9. User Effects, b_u

With user effects added to the model, the new RMSE is now 0.8663

- Predict with time effect considerations

```
time_avgs <- train_set %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(movie_avgs, by= "movieId") %>%
  left_join(user_avgs, by= "userId") %>%
  group_by(week) %>%
  summarize(ti = mean(rating - b_i - b_u - mu))

predicted_ratings_3 <- test_set %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(user_avgs, by = "userId") %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(time_avgs, by = "week") %>%
  mutate(prediction =mu+b_i+b_u+ti) %>%
  pull(prediction)
model_3_rmse <- RMSE(predicted_ratings_3, test_set$rating)
model_3_rmse
```

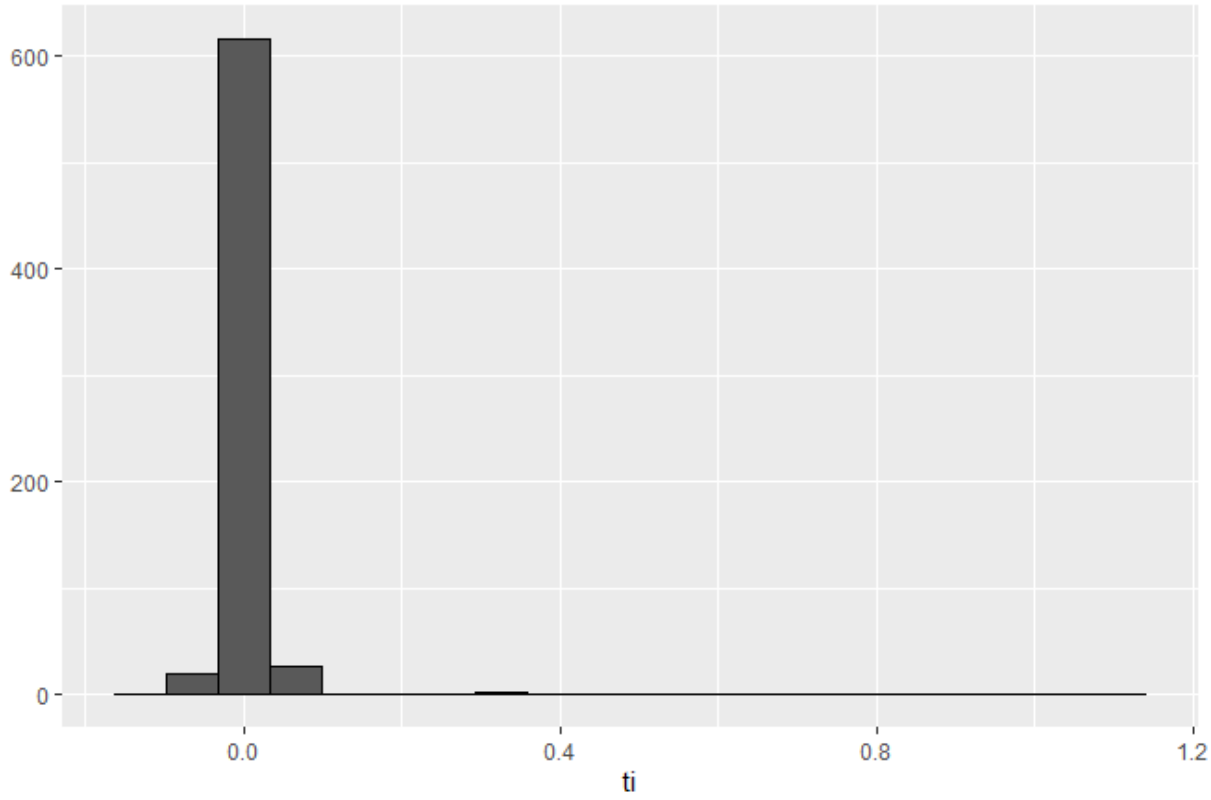

Figure 10: Fig 10. Time Effects, ti

By adding time of rating effect, the RMSE is very slightly improved to 0.8663

- Predict with genre effect consideration

```
genre_avgs <- train_set %>%
 mutate(week = round_date(rating_date, unit = "week")) %>%
 left_join(movie_avgs, by= "movieId") %>%
 left_join(user_avgs, by= "userId") %>%
 group_by(week) %>%
 left_join(time_avgs, by= "week") %>%
 ungroup() %>%
 group_by(genres) %>%
 summarize(ge = mean(rating - ti - b_i - b_u - mu))
predicted_ratings_4 <- test_set %>%
 mutate(week = round_date(rating_date, unit = "week")) %>%
 left_join(user_avgs, by = "userId") %>%
 left_join(movie_avgs, by = "movieId") %>%
 left_join(time_avgs, by = "week") %>%
 left_join(genre_avgs, by = "genres") %>%
 mutate(prediction =mu+b_i+b_u+ti+ge) %>%
 pull(prediction)
model_4_rmse <- RMSE(predicted_ratings_4, test_set$rating)
model_4_rmse
```

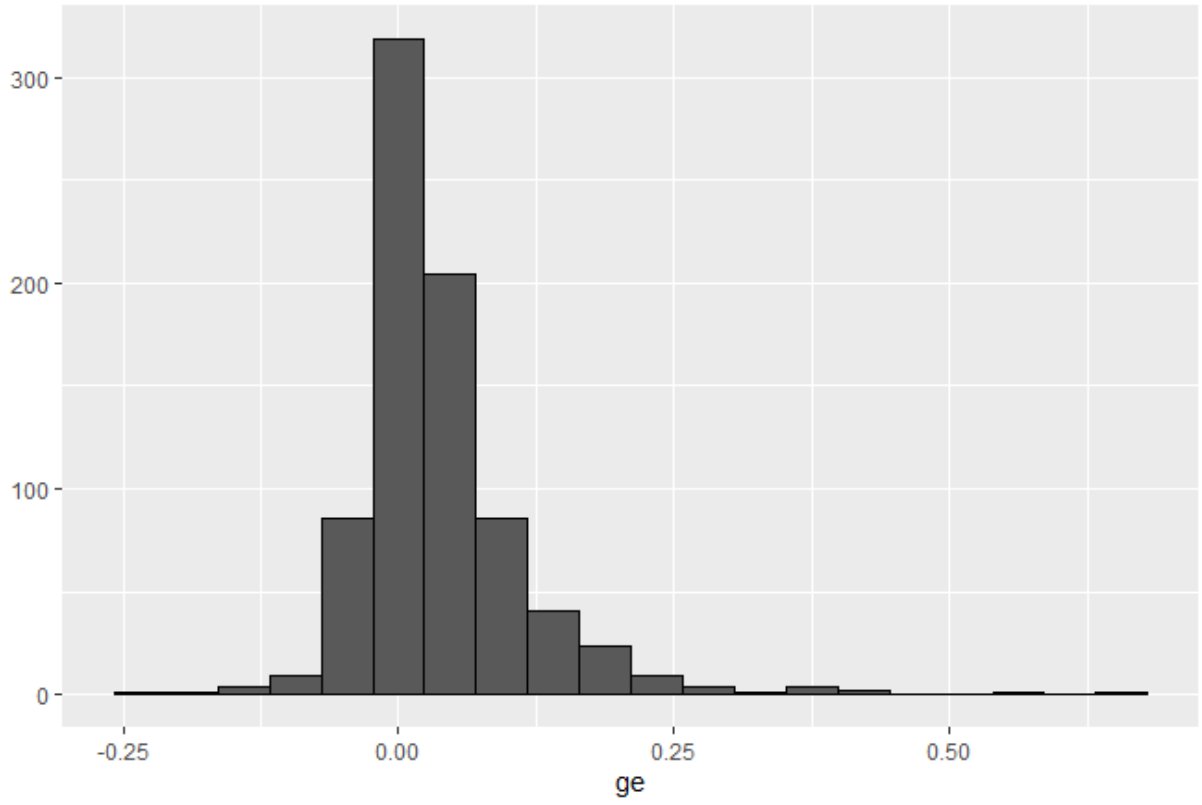

Figure 11: Fig 11. Genre Effects, ge

The genre effect provides a further reduction in the RMSE to 0.8659

- Predict using all the parameters above with regularization

```
l <- 5.5

b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))

b_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))

ti <- train_set %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(b_i, by= "movieId") %>%
  left_join(b_u, by= "userId") %>%
  group_by(week) %>%
  summarize(ti = sum(rating - b_i - b_u - mu)/(n()+l))

ge <- train_set %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(b_i, by= "movieId") %>%
  left_join(b_u, by= "userId") %>%
  group_by(week) %>%
  left_join(ti, by= "week") %>% ungroup() %>%
  group_by(genres) %>%
  summarize(ge = sum(rating - b_i - b_u - mu - ti)/(n()+l))


predicted_ratings_5 <- test_set %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(ge, by = "genres") %>%
  left_join(ti, by = "week") %>%
  mutate(pred = mu + b_i + b_u + ti + ge) %>%
  .$pred
model_5_rmse <-  RMSE(predicted_ratings_5, test_set$rating)
model_5_rmse
```

The regularization of all the features used in the model provides a significant improvement of RMSE to 0.8654. This model will now be evaluated on the Validation set that was held out.

- Applying the model to the validation set and calculating the final RMSE

```
final_ratings <-
  validation %>%
  mutate(week = round_date(rating_date, unit = "week")) %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(ge, by = "genres") %>%
  left_join(ti, by = "week") %>%
```

```
  mutate(pred = mu + b_i + b_u + ti + ge) %>%
  .$pred

final_rmse <- RMSE(final_ratings, validation$rating)
final_rmse
```

3. Results

Method	RMSE_result
Just Average	1.0603099
Movie Effect	0.9436226
User Effect	0.8663835
Time of Rating Effect	0.8663111
Genre Effect	0.8659941
Regularized Parameters	0.8654096
Final RMSE/Validation Set	0.8647396

Figure 12: Fig 12. RMSE Results Compilation

4. Conclusion

This project started off with large data set. The objective was to split the data set into a training set, a testing set, and a final evaluation set. A deep dive into the data set was done to study the structure of the data and its objects, exploratory data analysis was done to visualize patterns that may be helpful in building up the model. Finally, a model was build, taking into account the major features of the data set. The final model presented in this project achieves a RMSE value of 0.8647. This number proves satisfactory for the purpose of this project. However, it could be further improved by exploring other machine learning algorithms and considering other factors that were not explored in this project, e.g. the impact of the number of ratings that a user gives on a given day to the overall rating per user.