Movielens Recommendation System

Executive Summary

This report documents the analyses used to design the movie recommendation system based on the Movielens 10M dataset.

The Movielens 10M dataset was obtained from https://grouplens.org/datasets/movielens/10m/ and contained the following individual files -

1. ratings.dat - Each line represented one rating of one movie by one user. Movies and users are identified by movieId and userId respectively.
2. movies.dat - Each line represented one movie and provided titles and genres for each movieId

Prior to conducting analyses and building the recommendations system, the ratings.dat and movies.dat files were merged on movieId and split into training (edx) and test (validation) datasets such that the validation dataset contained 10% of all data.

Exploratory analyses were then conducted to get a feel for the data and assess plausibility of hypotheses.

A model based on the matrix factorization method was built on the edx set and tested on the validation set. Iterations were performed to enhance the accuracy of the model.

The best RMSE obtained on the validation set was 0.865.

Analysis

First step of the analysis is merging ratings.dat and movies.dat and creating the edx and validation datasets using the code given. The dimensions and structure of the edx and validation datasets created are as follows -

```
dim(edx)

## [1] 9000055       6

str(edx)

## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 83898
4474 838983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)"
"Stargate (1994)" ...
```

```
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|
Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...

dim(validation)

## [1] 999999      6

str(validation)

## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 86824
5920 1136075494 1133571200 844416936 844417070 ...
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alo
ne (1990)" "Rob Roy (1995)" ...
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|C
omedy" "Action|Drama|Romance|War" ...
```

Next, the edx dataset is checked for missing or NA values. The ratings field is also checked to confirm all values are within the required 0.5-5 range.

```
# Check if there are any blank values in the dataset
sapply(edx, function(x) sum(is.na(x)))

##    userId   movieId    rating timestamp     title    genres
##         0         0         0         0         0         0

# Check if any rating is > 5 or <= 0
sum(edx$rating > 5 | edx$rating <= 0)

## [1] 0
```

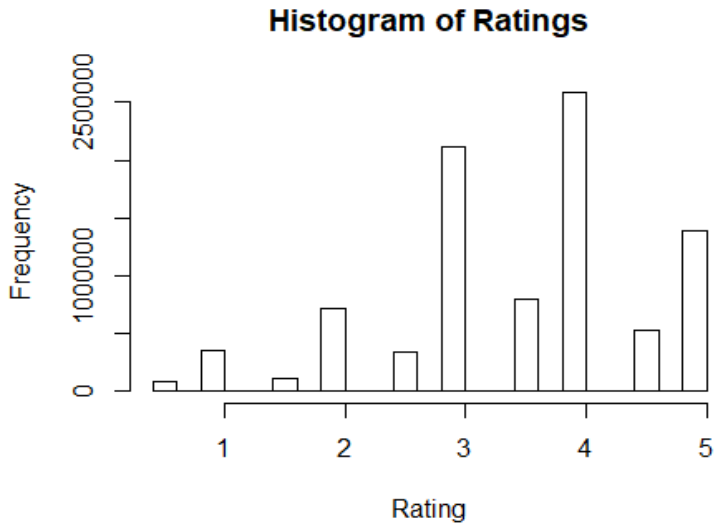
Descriptive statistics are reported / visualized. Specifically,

1. count of movies and users are calculated to get a sense of scope of the data

```
# Count of movies
length(unique(edx$movieId))

## [1] 10677

# Count of users
length(unique(edx$userId))

## [1] 69878
```

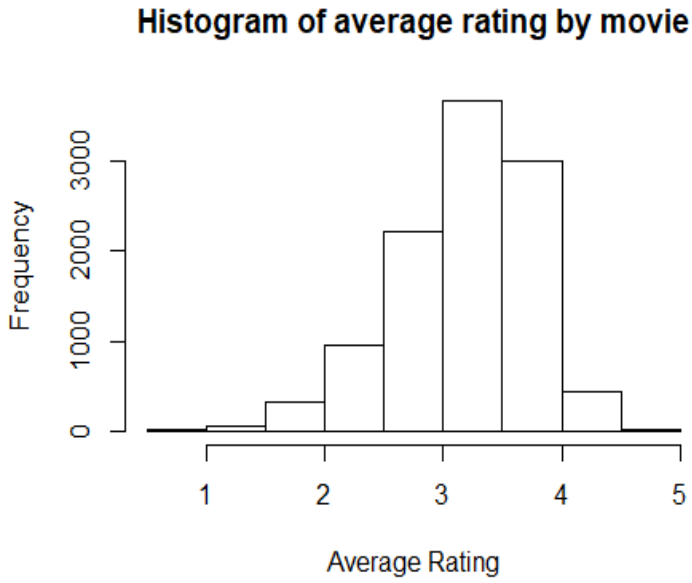
2. distributions of ratings are plotted to understand which ratings are most frequently given

```
# Distribution of ratings given
hist(edx$rating, xlab="Rating", main="Histogram of Ratings")
```

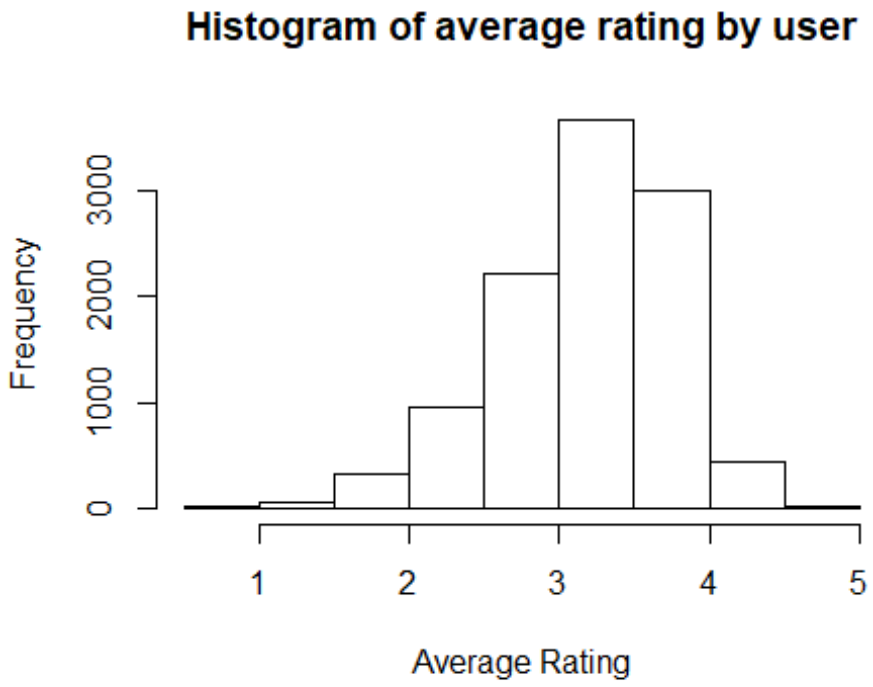


3. average ratings are computed for each movie and then plotted to see if different movies are rated differently. Similar analysis is done for users.

```
# Distribution of average ratings by movies
movie_avgs <- edx %>% group_by(movieId) %>% summarize(avg_rating = mean(ratin
g))
hist(movie_avgs$avg_rating, xlab="Average Rating", main="Histogram of average
rating by movie")
```



```
# Distribution of average ratings by users
user_avgs <- edx %>% group_by(userId) %>% summarize(avg_rating = mean(rating)
)
hist(movie_avgs$avg_rating, xlab="Average Rating", main="Histogram of average
rating by user")
```



Matrix Factorization based model is used to predict ratings in the validation set. The basic average rating model was first tried. Then, a movie bias, b_i, was built in to account for differences in ratings across movies. Finally, a user bias, b_u, was built in to account for differences in ratings given by different users.

```
# Just the average rating across all movies and users
mu <- mean(edx$rating)
mu

## [1] 3.512465

# Account for differences between movies
movie_avgs <- edx %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu)
)
pred_movies <- mu + validation %>% left_join(movie_avgs, by='movieId') %>% .$
b_i

# Account for differences between users
user_avgs <-
  edx %>% left_join(movie_avgs, by='movieId') %>%
```

```
  group_by(userId) %>% summarize(b_u = mean(rating - mu - b_i))

pred_users <-
  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>% .$pred
```

Results

In the data preprocessing stage, it was observed that the data is accurate and no cleaning, treatment or imputation is necessary.

Looking at the distribution of all ratings, it was observed that 4 was the most frequently given rating followed by 3. It was also observed that half star ratings are less common than full star ratings.

Looking at the distribution of average rating by movie, it is evident that there are substantial differences between ratings for different movies. This made the case for building in a movie bias in the model.

Looking at the distribution of average rating by user, it is evident that different users tend to rate movies differently. Some consistently give 5 star ratings, whereas some give much lower ratings on average. This made the case for building in a user bias.

Root mean squared error (RMSE) was calculated for the three models.

In the first model, the simple average rating across all movies in the edx set was used as prediction for ratings in the validation set. This resulted in an RMSE of 1.06.

```
naive_rmse <- RMSE(mu, validation$rating)
naive_rmse

## [1] 1.061202
```

In the second model, the bias term b_i to account for movie differences was calculated and added to the naive average mu. This reduced the RMSE to 0.94.

```
model_1_rmse <- RMSE(pred_movies, validation$rating)
model_1_rmse

## [1] 0.9439087
```

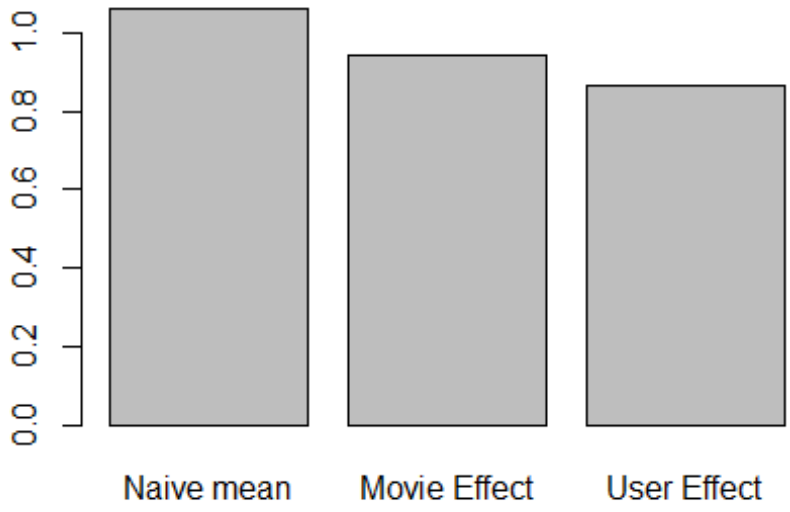
Finally, the bias term b_u to account for user differences was calculated and added to mu + b_i. This further reduced the RMSE to 0.865, thus achieving the desired result.

```
model_2_rmse <- RMSE(pred_users, validation$rating)
model_2_rmse
```

```
## [1] 0.8653488
```

Here's a plot of the three RMSE values to visualize the model improvement at each stage.

```
barplot(c(naive_rmse, model_1_rmse, model_2_rmse), names.arg = c("Naive mean"
, "Movie Effect", "User Effect"))
```



Conclusion

A movie recommendation system was designed and built using the matrix factorization method on the 10M movielens dataset.

The best RMSE was achieved after accounting for movie bias and user bias. The fact that RMSE decreased each time when movie bias was introduced and when user bias was introduced clearly suggests that the movie that is being rated and the user who is rating it have a significant influence on the given rating. Therefore, any movie recommendation model should include movie and user identifiers as predictor variables.

Additionally, techniques like multiclass classification models, collaborative filtering, similarity mesaures, etc. can be tested to iterate and improve the performance of the recommendation system.