Capstone: MovieLens Project Report

Michael G. Mielach

02-01-2023

Introduction

A recommender system, also known as a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. The basic idea behind recommender systems is to find patterns in data and use them to predict users' interests in order to provide personalized recommendations. This is done by searching for patterns in data that can be used to predict which items a user might like, and then presenting those items to the user.

For the "MovieLens Project" we use a 10% slice from the 10M version of the MovieLens dataset. The goal of this project is to develop a recommendation system using machine learning techniques. The system will be trained on inputs from a training set ("edx") to predict movie ratings in the validation set ("final_holdout_test") . The root mean squared error (RMSE) will be used to evaluate how closely the predictions match the true values in the validation set. The project will do an exploratory study of the data to identify potential patterns first and analyze the performance of various machine learning techniques with a focus on regression models afterwards. The objective is to identify the recommendation system with the best performance.

Methods/Analysis

section that explains the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and your modeling approach

1) Dataset preparation:

The data set is loaded from the grouplens which split the data into edx set and 10% validation set. the edx set will be split into training and test set,and validation set will be used to final evaluation.

```
library(tidyverse)
library(caret)

options(timeout = 120)

#load data

    dl <- "ml-10M100K.zip"
    download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

    ratings_file <- "ml-10M100K/ratings.dat"
    if(!file.exists(ratings_file))
      unzip(dl, ratings_file)

    movies_file <- "ml-10M100K/movies.dat"
    if(!file.exists(movies_file))
      unzip(dl, movies_file)
```

```
# Create data frame

    ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),stringsAs
    colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
    ratings <- ratings %>%
      mutate(userId = as.integer(userId),
             movieId = as.integer(movieId),
             rating = as.numeric(rating),
             timestamp = as.integer(timestamp))

    movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                            stringsAsFactors = FALSE)
    colnames(movies) <- c("movieId", "title", "genres")
    movies <- movies %>%
      mutate(movieId = as.integer(movieId))

    movielens <- left_join(ratings, movies, by = "movieId")


# Create final hold-out/validation test set

    # Slice 10% out of MovieLens data

        set.seed(1, sample.kind="Rounding")

        test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
        edx <- movielens[-test_index,]
        temp <- movielens[test_index,]

    # Make sure userId and movieId in final hold-out/validation test set are also in edx set

        final_holdout_test <- temp %>%
          semi_join(edx, by = "movieId") %>%
          semi_join(edx, by = "userId")

    # Add rows removed from final hold-out/validation test set back into edx set

      removed <- anti_join(temp, final_holdout_test)
      edx <- rbind(edx, removed)

    # remove some items which are no longer required

      rm(dl, ratings, movies, test_index, temp, movielens, removed)

    # Check for any NA value

      anyNA(edx)
```

2) Exploratory Data Analysis:

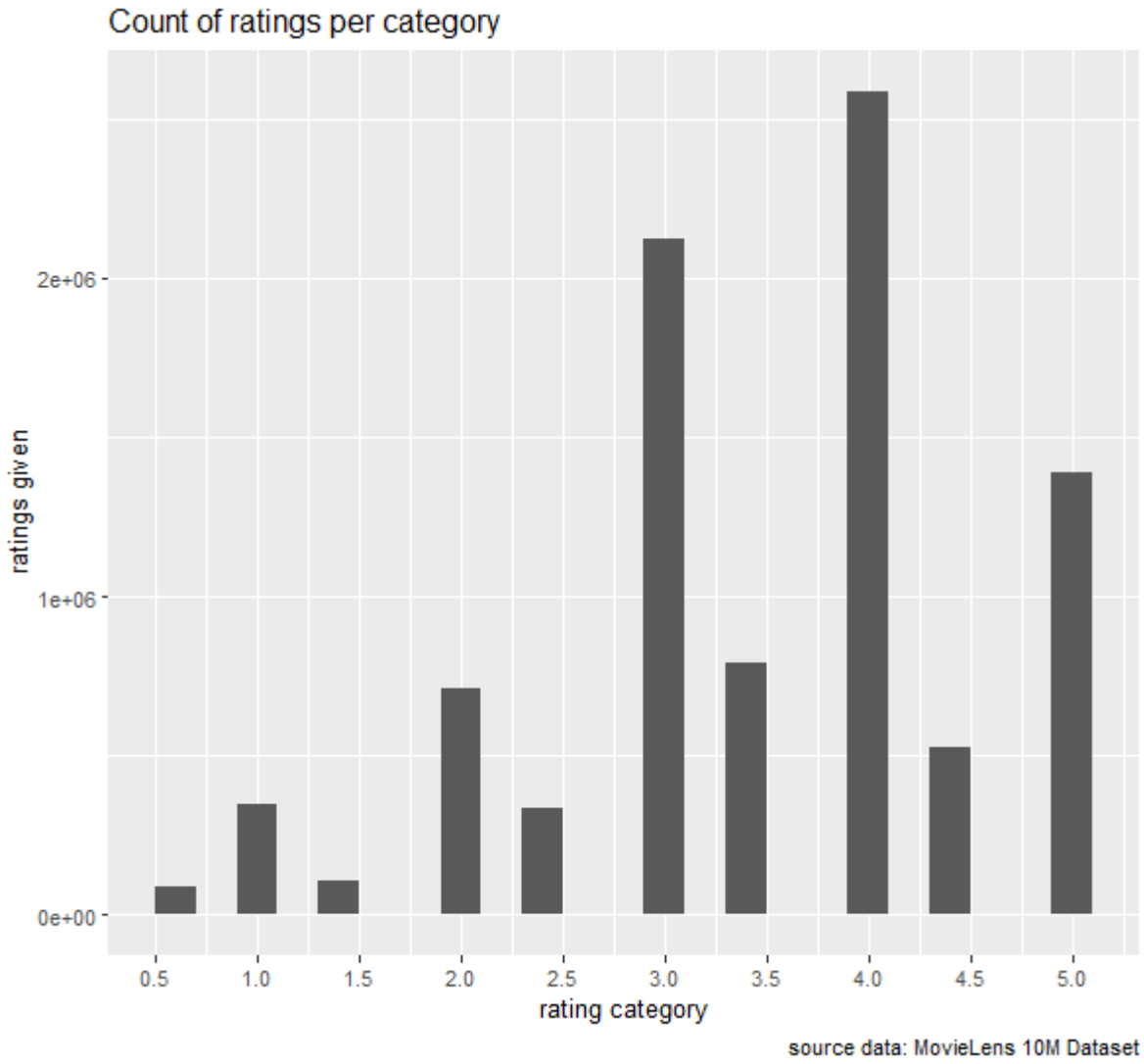
Quick summary of the dataset & validation set

Code:

```
#Analysis of the data structure (test & validation set):

    summary(edx)
    str(edx)

    summary(final_holdout_test)
    str(final_holdout_test)

    edx %>% summarize(unique_users = length(unique(userId)),
                      unique_movies = length(unique(movieId)),
                      unique_genres = length(unique(genres)),
                      unique_rating = length(unique(rating)))
```

The edX dataset has 6 features and is made up of around 9,000,055 observations. The validation set, which is 10% of the 10M Movielens dataset, contains the same features as the edX set, but has 999,999 observations. The userId and movieId in the edX set are also present in the validation set. Each row in the dataset represents a rating given by a user to a movie. The "rating" column is the outcome that we want to predict. The dataset contains 69,878 unique users, 10,677 unique movies and 797 unique genres. Users can give ratings ranging from 0.5 to 5.0 being the highest resulting in 10 different rating scores.

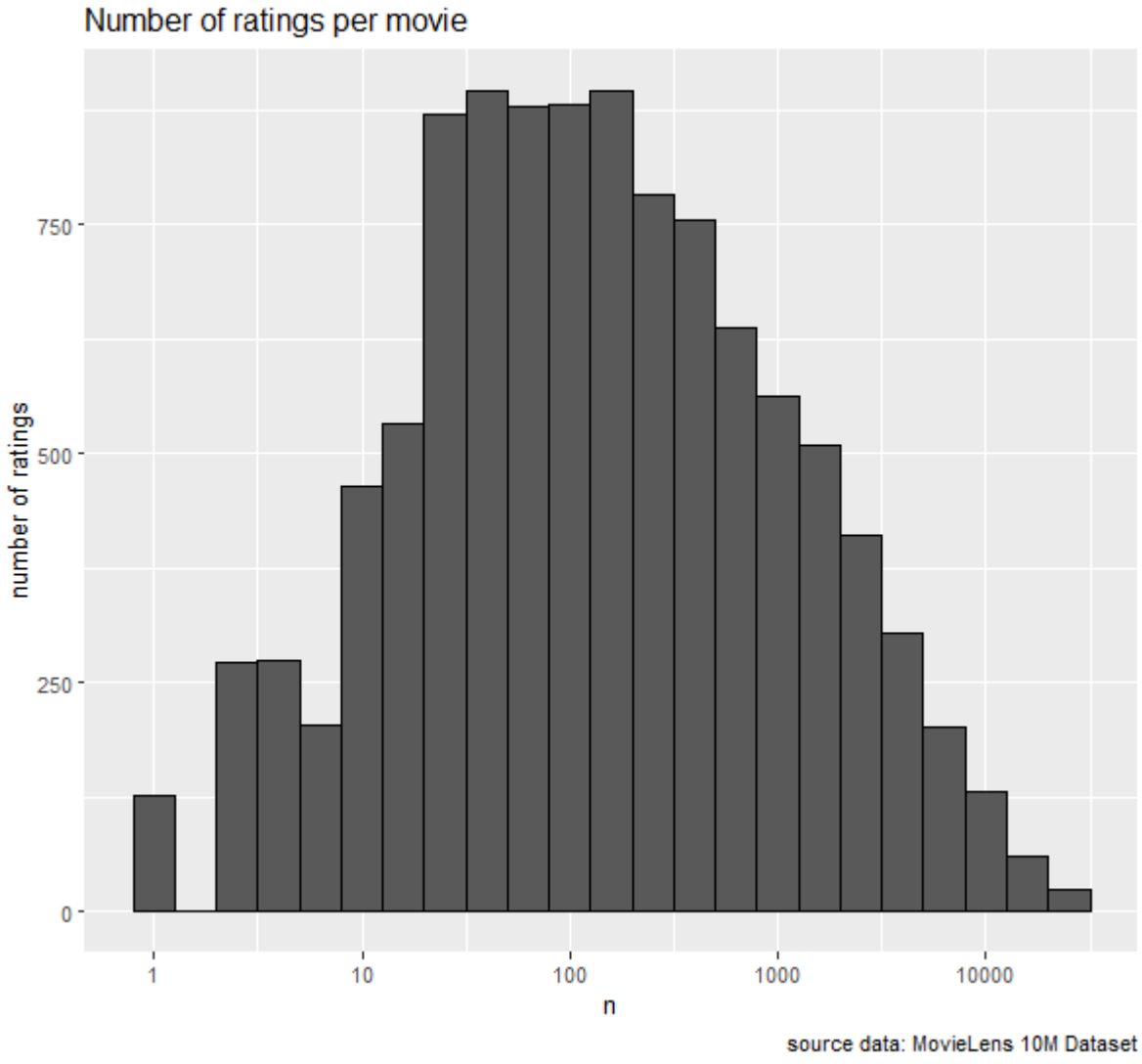
```
#Analysing rating data

    range(edx$rating)

    edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>%
      arrange(desc(count))

    edx %>%
      ggplot(aes(x= rating)) +
      geom_histogram(bins = 10, binwidth = 0.2) +
      scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
      labs(x="rating category", y="ratings given", caption = "source data: MovieLens 10M Dataset") +
      ggtitle("Count of ratings per category")
```

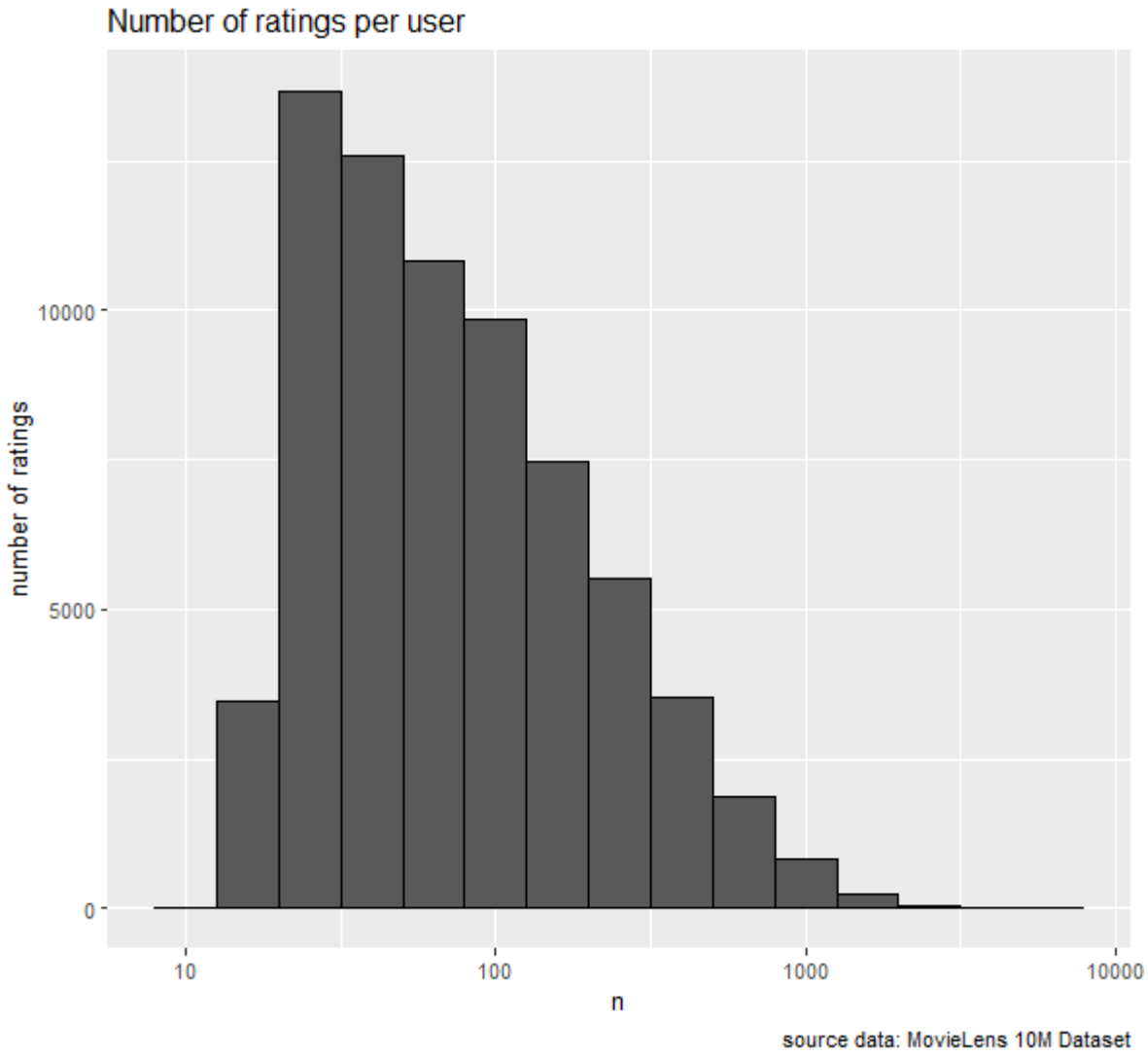


The top 5 ratings, in order from most to least common, are: 4, 3, 5, 3.5, and 2. The histogram indicates that ratings with half stars are less frequent than ratings with whole stars.

```
# histogram of number of ratings by movieId

    edx %>%
      count(movieId) %>%
      ggplot(aes(n)) +
      geom_histogram(bins = 30, binwidth=0.2, color="black", ) +
      scale_x_log10() +
      labs(x="n", y="number of ratings", caption = "source data: MovieLens 10M Dataset") +
      ggtitle("Number of ratings per movie")


# histogram of number of ratings by userId

    edx %>%
      count(userId) %>%
      ggplot(aes(n)) +
```

```
geom_histogram(bins = 30, binwidth=0.2, color="black", ) +
scale_x_log10() +
labs(x="n", y="number of ratings", caption = "source data: MovieLens 10M Dataset") +
ggtitle("Number of ratings per user")
```





Exploring the number of ratings by movieId and by userId through visualization shows that some movies receive more ratings than others, and some users are more active in rating movies.This likely explains the presence of movie effects and user effects in the data.

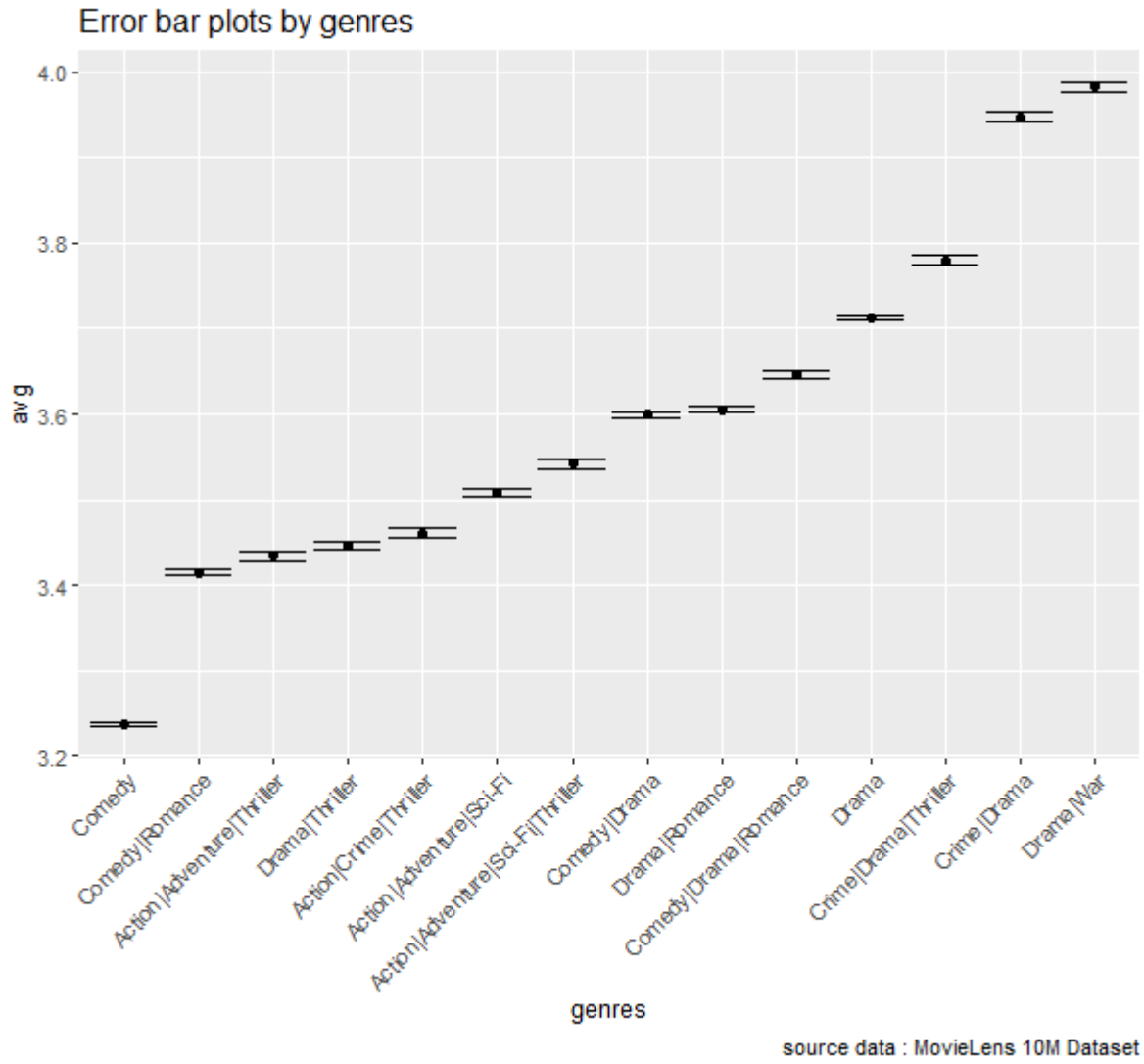
```
#inspection of titles

    edx %>%
      group_by(title) %>%
      summarize(count=n(), rating = mean(rating)) %>%
      top_n(10,count) %>%
      arrange(desc(count))
```

```
# A tibble: 10 x 3
   title                                                     count rating
   <chr>                                                     <int>  <dbl>
 1 Pulp Fiction (1994)                                       31362   4.15
 2 Forrest Gump (1994)                                       31079   4.01
 3 Silence of the Lambs, The (1991)                          30382   4.20
 4 Jurassic Park (1993)                                      29360   3.66
 5 Shawshank Redemption, The (1994)                          28015   4.46
 6 Braveheart (1995)                                         26212   4.08
 7 Fugitive, The (1993)                                      25998   4.01
 8 Terminator 2: Judgment Day (1991)                         25984   3.93
 9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672   4.22
10 Apollo 13 (1995)                                          24284   3.89
```

If the movie titles are ordered according to their count of ratings it can be seen that the most rated movies are among the most popular as well.

```
#inspection of genre

    edx %>% group_by(genres) %>%
      summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
      filter(n >= 100000) %>%
      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 = 45, hjust = 1)) +
      labels(title = "Error bar plots by genres" , caption = "source data : MovieLens 10M Dataset")


    edx %>% separate_rows(genres, sep = "\\|") %>%
      group_by(genres) %>%
      summarize(count = n()) %>%
      arrange(desc(count))
```

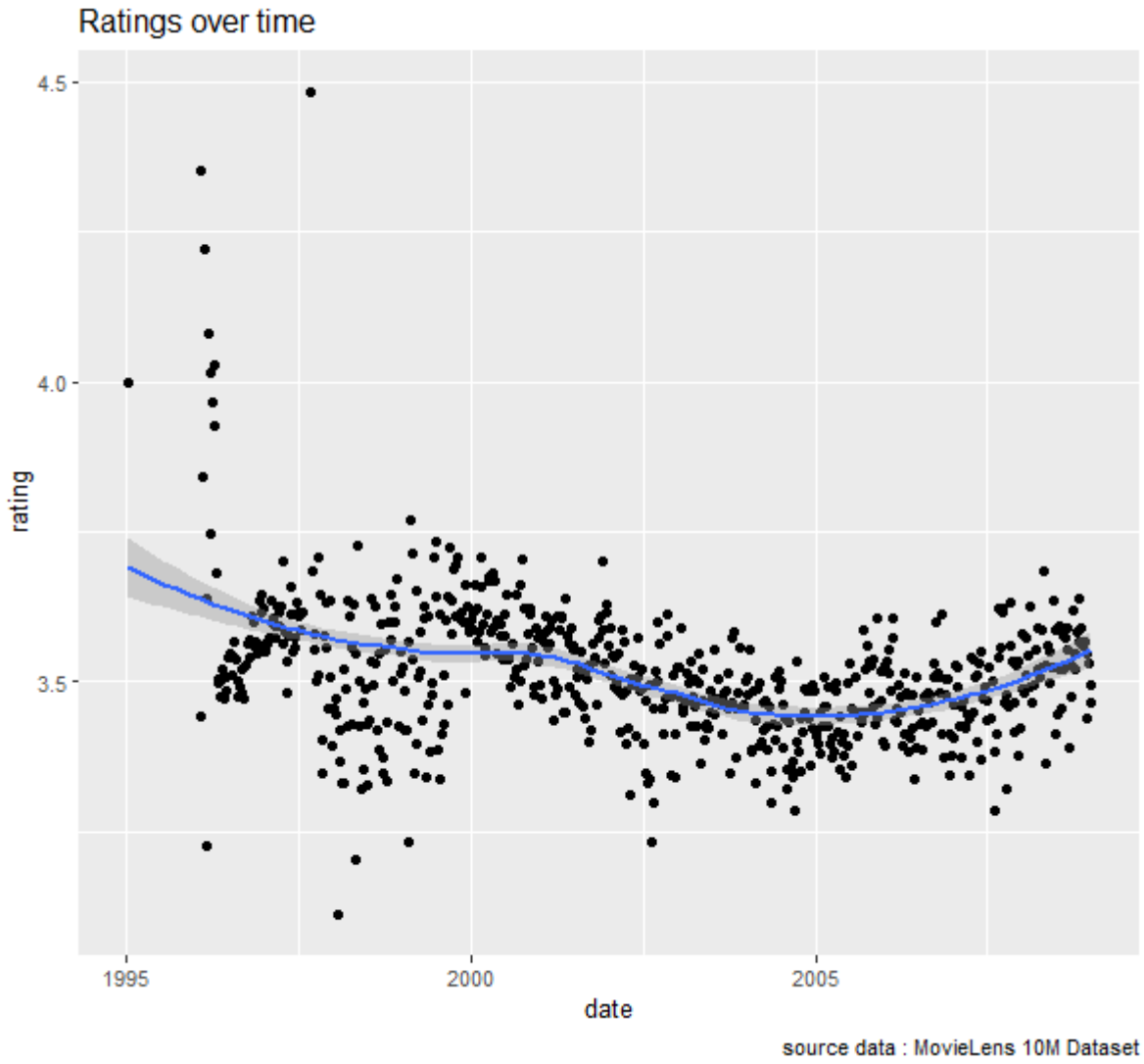
```
# A tibble: 20 x 2
   genres               count
   <chr>                <int>
 1 Drama              3910127
 2 Comedy             3540930
 3 Action             2560545
 4 Thriller           2325899
 5 Adventure          1908892
 6 Romance            1712100
 7 Sci-Fi             1341183
 8 Crime              1327715
 9 Fantasy             925637
10 Children            737994
11 Horror              691485
12 Mystery             568332
13 War                 511147
14 Animation           467168
15 Musical             433080
16 Western             189394
17 Film-Noir           118541
18 Documentary          93066
19 IMAX                 8181
20 (no genres listed)      7
```



Looking at the genre data the most frequently rated are Drama and Comedy as well as Action and Thriller. The error bar chart shows evidence of a genre effect based on the different average ratings per genre from Comedy with the lowest average rating up to Drama/War with the highest average rating.

```
#inspection of time effects
```

```
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Ratings over time")+
  labs(caption = "source data : MovieLens 10M Dataset")
```



Looking at the data over time it appears that the trend of average ratings is drifting slightly. There may be some evidence of a time effect, but it is not particularly strong.

The exploratory data analysis gives a good first understanding to better choose and adjust the prediction models to create and test the recommendation system. In summary it shows that there seem to be effects on ratings from users, movie titles and genres. The variation over time seems rather negligible. Therefore I focus on linear models taking those effects into account in the following chapter.

3) Prediction Modelling:

This section explains the used models and their performance evaluation. Three models (model 2a, 2b and 3) are regression models which can be used as a simple approach to create a recommendation system. To evaluate their performance they are benchmarked against a baseline (model 1). The key metric to evaluate model performance is the root mean square error (RMSE) reflecting the difference between the actual rating values in the validation set and the rating predictions made by the used models (see general formula below). A lower RMSE usually indicates a better model performance for making predictions which are closer to the actuals (but one has to be aware of the unwanted outcome of overfitting models).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2}$$

Model 1 (Baseline Model)

In the first model I assume that all movies receive the same rating and do not take into account any biases. I predict a new rating to be the average rating of all movies in our training dataset. Therefore the prediction for all movies in the validation set is the mean movie rating of 3.51. The resulting RMSE is 1.061.

```
#Model 1 (Baseline Model):

    mu <- mean(edx$rating)
    model_baseline_rmse <- RMSE(mu,final_holdout_test$rating)
```

Model 2 (Multivariate Linear Regression Models)

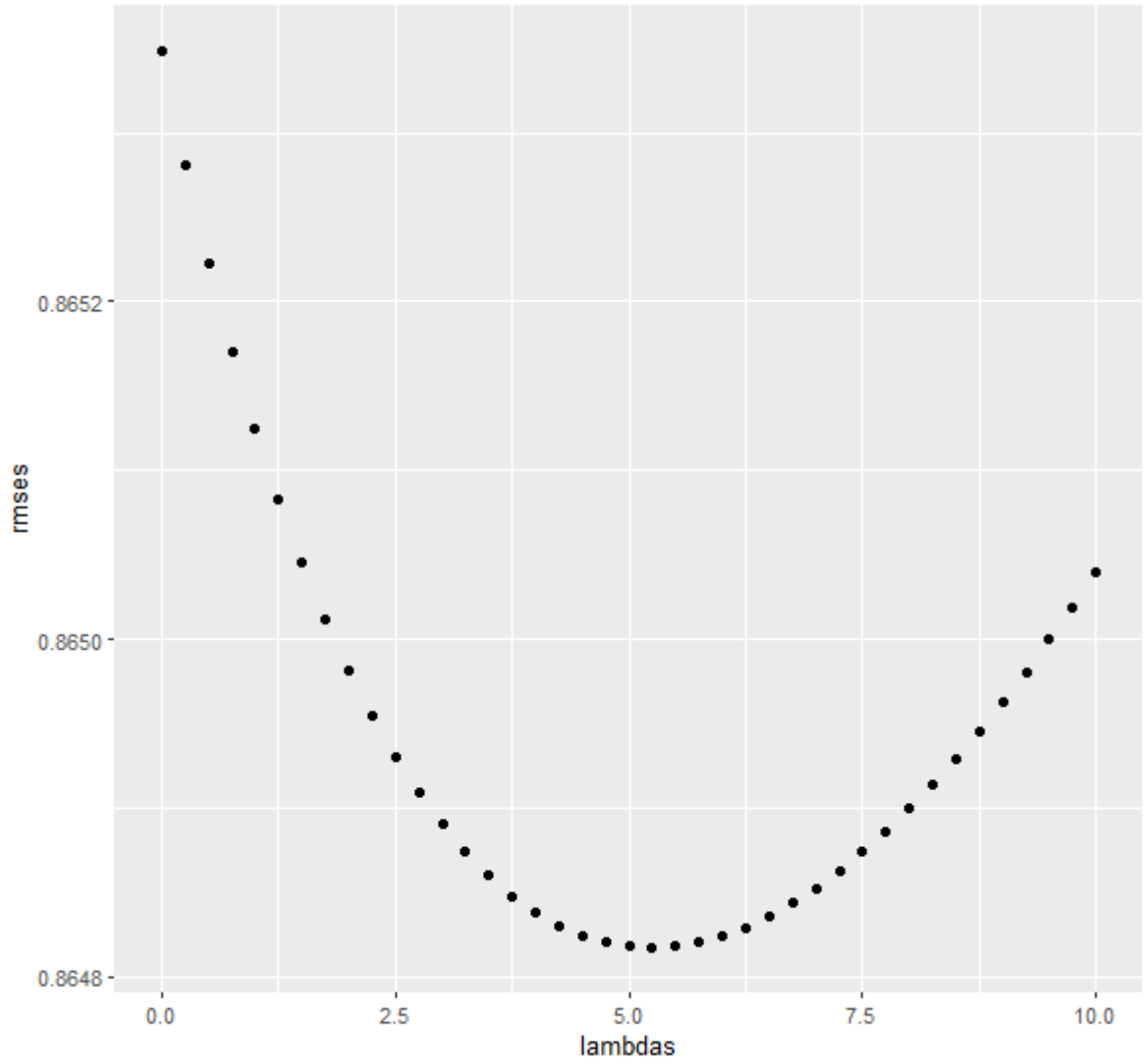
The second model takes into account movie and user effects (model 2a) and genre effects (model 2b). Based on the exploratory data analysis it seems that a certain amount of rating variation can be explained by those factors. The results of both models (2a and 2b) show the RMSE significantly improves compared to only using the average as a predictor (model 1) to a RMSE of 0.8653. Adding the genre effect only slightly improves RMSE compared to only using movie and user effects to 0.8649 accordingly.

```
#Model 2 (Multivariate Regression Models):

#Model 2a (Movie and user effects):

    #Calculate movie effect bias

    b_i <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = mean(rating - mu))

    #Calculate user effect bias

    b_u <- edx %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = mean(rating - b_i - mu))

    #predict ratings using movie and user bias

    predicted_ratings_m2a <-
      final_holdout_test %>%
      left_join(b_i, by = "movieId") %>%
      left_join(b_u, by = "userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      pull(pred)

    #Calculate RMSE and improvement ratio

    model_2a_rmse <- RMSE(predicted_ratings_m2a, final_holdout_test$rating)
    model_2a_rmse
```

```
   impr_2a <- model_2a_rmse/model_baseline_rmse-1


#Model 2b (Movie, User and Genre effects):

   #Calculate genre effect bias

   b_g <- edx %>%
     left_join(b_i, by='movieId') %>%
     left_join(b_u, by='userId') %>%
     group_by(genres) %>%
     summarize(b_g = mean(rating - mu - b_i - b_u))

   #Add genre effect to predict ratings in the training set

   predicted_ratings_m2b <-
     final_holdout_test %>%
     left_join(b_i, by = "movieId") %>%
     left_join(b_u, by = "userId") %>%
     left_join(b_g, by = "genres") %>%
     mutate(pred = mu + b_i + b_u + b_g) %>%
     pull(pred)

   #Calculate RMSE and improvement ratio

   model_2b_rmse <- RMSE(predicted_ratings_m2b, final_holdout_test$rating)
   model_2b_rmse

   impr_2b <- model_2b_rmse/model_2a_rmse-1
```

Model 3 (Penalized Least Squares Linear Regression Model)

The third model is a more sophisticated approach to address or rather penalize outliers. As it can be seen from the data analysis above a lot of uncertainty/outliers could come from less rated movies and rather inactive users. Therefore this approach uses an optimization parameter (lambda) to minimize RMSE by penalizing outliers. To find the optimal lambda we create a function of lambda based on the model 2a from the previous chapter to look for the RMSE minimizing lambda in a following step. As can be seen from the data the optimal lambda is 5.25 (see chart below) resulting in a RMSE of 0.8648.




```
#Model 3 (Penalized Least Squares)

#Extending the model from 2a with the optimization parameter lambda

    lambdas <- seq(0, 10, 0.25)

    rmses <- sapply(lambdas, function(l){

      #Calculate the mean of ratings from the edx training set
      mu_reg <- mean(edx$rating)

      #Adjust mean by movie effect and penalize low number on ratings
      b_i_reg <- edx %>%
        group_by(movieId) %>%
        summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))

      #Adjust mean by user and movie effect and penalize low number of ratings
      b_u_reg <- edx %>%
        left_join(b_i_reg, by="movieId") %>%
```

```
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))

  #predict ratings in the training set to derive optimal penalty value 'lambda'
  predicted_ratings_m3 <-
    final_holdout_test %>%
    left_join(b_i_reg, by = "movieId") %>%
    left_join(b_u_reg, by = "userId") %>%
    mutate(pred = mu_reg + b_i_reg + b_u_reg) %>%
    pull(pred)

  return(RMSE(final_holdout_test$rating,predicted_ratings_m3))
})


#Optimize lambda factor (to minimize RMSE)

qplot(lambdas, rmses)

lambda <- lambdas[which.min(rmses)]
lambda

#Calculate RMSE and improvement ratio

model_3_rmse <- min(rmses)
model_3_rmse

impr_3 <- model_3_rmse/model_2b_rmse-1
```

After applying the above mentioned models as recommender systems and evaluation their performance the next chapter compares and sums up the results.

Results

To finally compare all the results a table aggregating the results and the respective improvements to the previous model is created (see table below). As mentioned before the linear regression model using movie and user effects already lead to a major improvement of the RMSE which subsequently could be further optimized by accounting for the genre effect and with the help of a penalizing factor for outliers. In sum the recommendation system using the penalized least squares approach leads to the best performance of all models with a RMSE of 0.8648.

Method	RMSE	Improvement
Model 1: Baseline	1.0612018	0.0000000
Model 2a: Movie + User Effects	0.8653488	-0.1845577
Model 2b: Movie + User + Genre Effects	0.8649469	-0.0004645
Model 3: Penalized least squares	0.8648170	-0.0001502

```
#Comparing Results

    rmse_results <- data_frame(Method = c("Model 1: Baseline",
                                          "Model 2a: Movie + User Effects",
                                          "Model 2b: Movie + User + Genre Effects",
                                          "Model 3: Penalized least squares"),
                               RMSE = c(model_baseline_rmse,
```

```
                                    model_2a_rmse,
                                    model_2b_rmse,
                                    model_3_rmse),
                          Improvement = c(0,
                                          impr_2a,
                                          impr_2b,
                                          impr_3))

rmse_results %>% knitr::kable()
```

Conclusion

The objective of the data science Capstone project was to create a movie recommendations system using the MovieLens dataset. After the movielens dataset is split into a test set and a validation set a closer look is taken on the test set data. The exploratory data analysis indicates that movie, user and potentially genre effects are the most promising effects in accurately predicting movie ratings. Plotting ratings against the user and movie data it becomes apparent that the numbers of ratings is not evenly spread across all movies and users. Consequently three different regression models are evaluated based on their RMSE (root mean squared error) as a recommendation system vs. just using the average movie rating as a predictor. The first model uses movie and user effects to predict movie ratings, the second model adds a genre effect to the two previously mentioned factors. The third penalized least squares model accounts for the concentration of ratings across movies and users (or rather the uncertainty which comes from less rated movies or less active users) through regularization. Evaluating the performance of the three models the RMSE improved from 1.060 to 0.8648 showing that using linear regression models as recommendation systems improves performance by around 18.4% (compared to the baseline), with the regularized linear regression model having the best performance. The analysis of this report can be extended to more sophisticated prediction models in future work to evaluated their performance improvement potential.