(HarvardX PH125.9x)
Data Science: Capstone

Capstone Project

Build a movie recommendation system

February 2022

submitted by Tobias Sveaas

Contents

Introduction

In this project we develop a recommendation system that predicts the rating a specific user will give a specific movie in terms of stars. The star rating system ranges from the lowest rating possible of 0.5 stars meaning the user would hate the movie to the highest possible rating of 5 stars meaning the user would love the movie.

The data used in this project is the MovieLens 10m movie ratings data set which can be found here https://grouplens.org/datasets/movielens/10m/ . The data set contains 10 million movie ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released in 2009.

The objective of this project is to train a model that achieves a Root Mean Squared Error (RMSE) below 0.86499 when predicting the movie ratings on the validation set. For this purpose a linear model was developed which attempts the convey the movie, user and genre specific effects on the rating. These effects were also regularized to account for the movies, users and genres with small sample sizes. Lastly matrix factorization was added on to account for the remaining variability in ratings . This model achieved an RMSE of .7946383 which fell well below the target value.

RMSE is a loss function that is defined as follows:

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

Where N is the numbers of ratings, u,i is the rating given to the movie i by user u and y_{u,i} is the predicted rating from the model of movie i by user u. This will evaluate how close our predictions are to the observed ratings.

Load packages and libraries needed in the project

First we install and load in the packages and libraries we will be using for the project

```
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(ggthemes)
library(lubridate)
library(scales)
library(recosystem)
```

Data cleaning

We load in the MovieLens data set and split it into two sets. The edx set (90% of the data) which will be used for the process of creating our model and the final validation set (10% of the data) that is used only for the final test of our chosen model.

Data Exploration

Exploratory analysis is done on the data set to get an overview of the main characteristics of the data set.

First we will will examine the dimensions of the edx set

x
9000055
6

The edx data set set contains 9,000,055 rows and 6 columns

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

The 6 variables in the data set are userId, movieId, rating, timestamp, title and genres

n_users	n_movies
69878	10677

There are 69,878 unique users and 10,677 unique movies in the data set

distinct_ratings	rating_range
10	0.5
10	5.0

There are 10 unique ratings that range from 0.5 to 5.0

distinct_genres
797

There are 797 unique genres

x
Comedy|Romance
Action|Crime|Thriller
Action|Drama|Sci-Fi|Thriller
Action|Adventure|Sci-Fi
Action|Adventure|Drama|Sci-Fi
Children|Comedy|Fantasy

There are so many unique genres because they consist of combinations of genres

```
## [1] 14
```

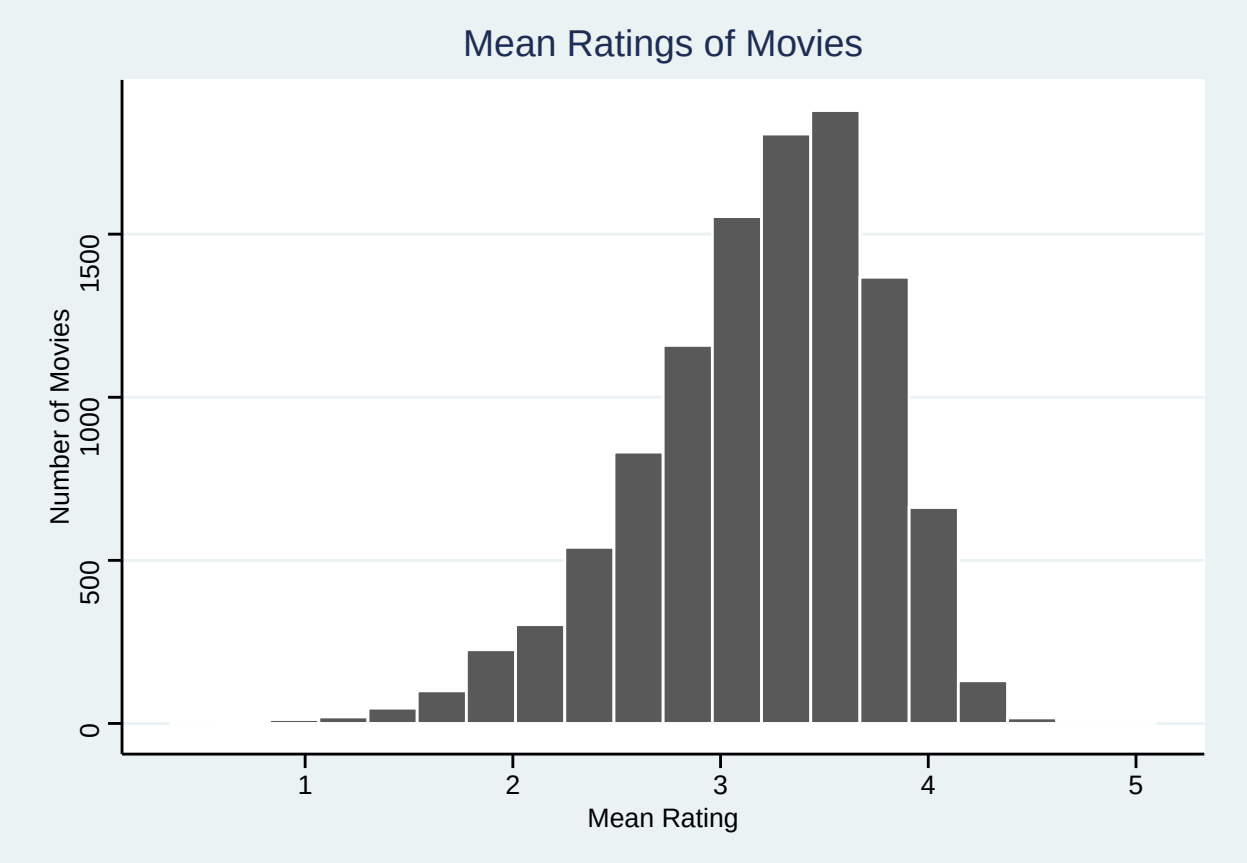
The ratings span over a period of 14 years

Data Analysis and Visualisation

Here we further analyse the data through visualization to explore trends and patterns that may have an effect on the ratings of movies. To find out about which variables could be useful to use in our model

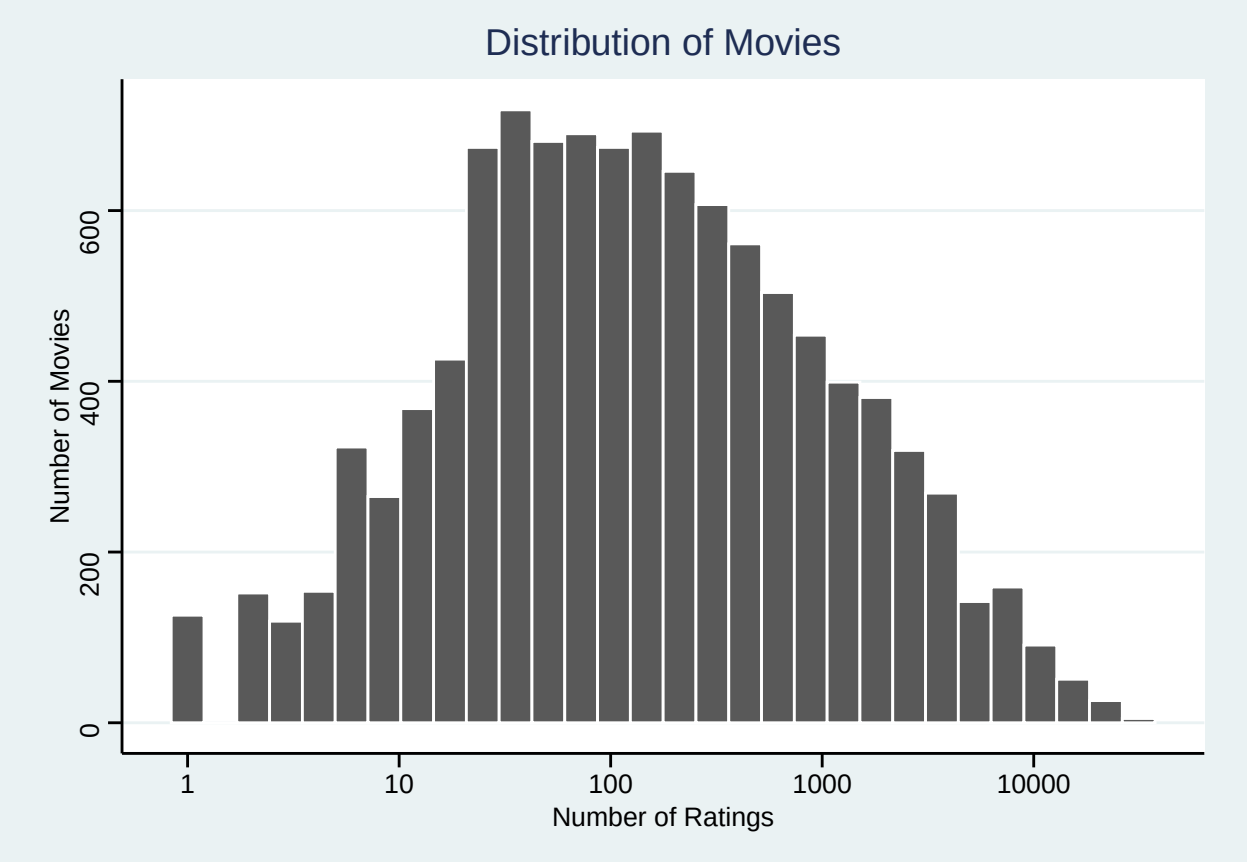
Movie Data

Lets find out the mean ratings for each movie to see different movies have noticeably different ratings



We see that there is quite a spread in average rating per movie in that some movies are rated higher than others.

Lets see the number of ratings per movie to see if each movie is rated a similar number of times



We see that there is a vast difference in the number of ratings per movie with some movies with over 10,000 ratings and some movies with under 10.

Lets see for the extremes of the mean rating graph how many times the movies were rated to see if the variability can be explained partially due to a small amount of ratings.

title	mean_rating	count
Blue Light, The (Das Blaue Licht) (1932)	5	1
Fighting Elegy (Kenka erejii) (1966)	5	1
Hellhounds on My Trail (1999)	5	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	5	2
Shadows of Forgotten Ancestors (1964)	5	1

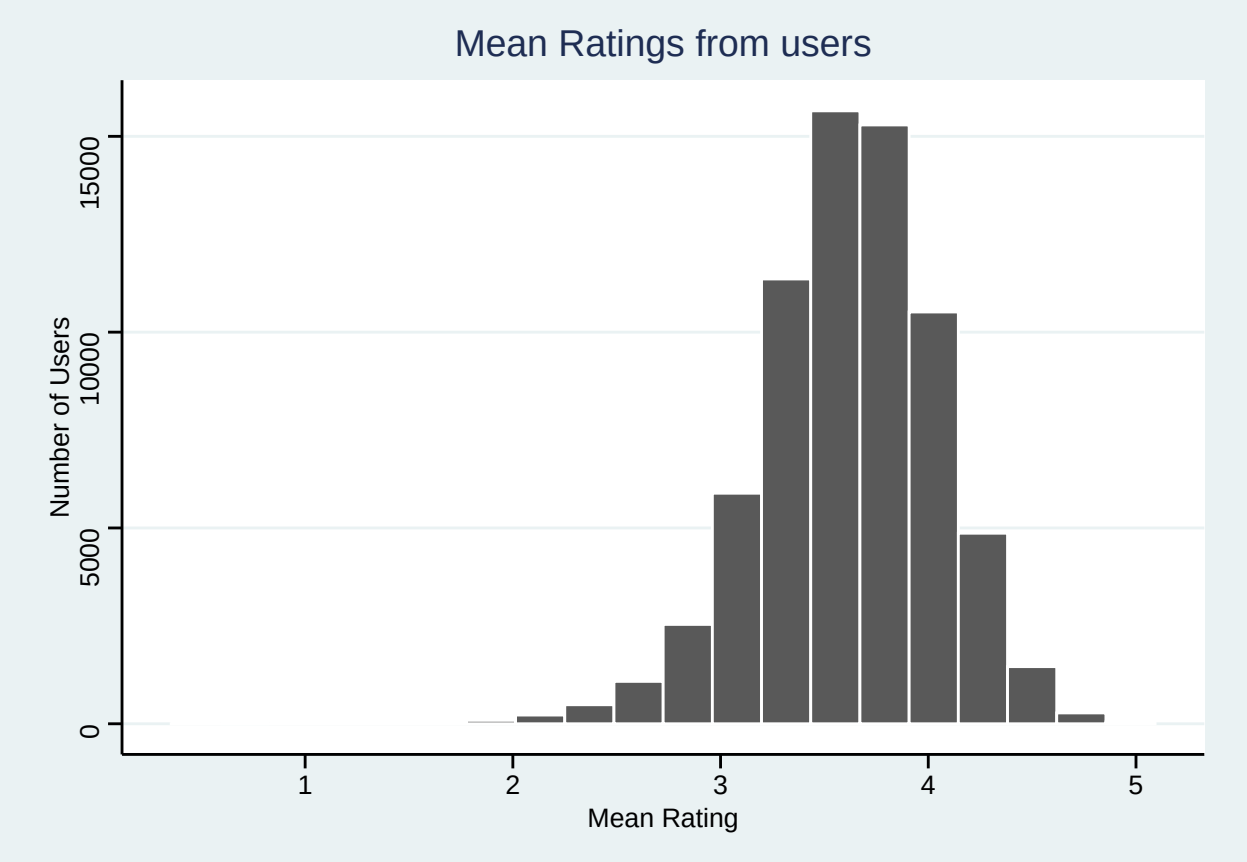
title	mean_rating	count
Accused (Anklaget) (2005)	0.5	1
Besotted (2001)	0.5	2
Confessions of a Superhero (2007)	0.5	1
Hi-Line, The (1999)	0.5	1
War of the Worlds 2: The Next Wave (2008)	0.5	2

We do indeed see that the the top 5 and bottom 5 rated movies are very obscure movies with only 1 or 2 ratings. So when creating our model we will have to include regularization to account for the high variability of movies with less ratings.

User Data

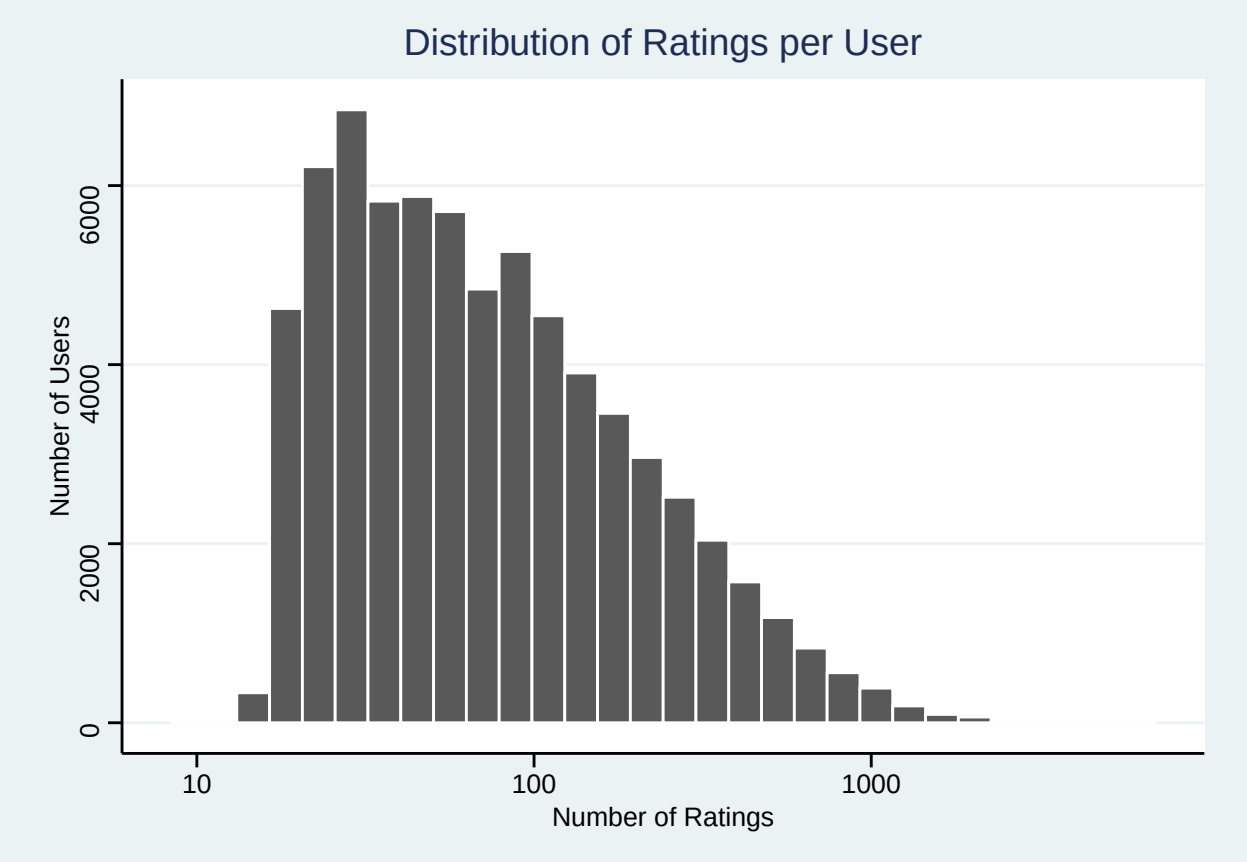
Lets find out if different users rate movies differently. As some may be be more favorable rating every movie highly and some may be critical rating every movie lower on average.

First lets plot a histogram of average rating from different users



We can see that although the spread is not a great as the effect different movies have there does still seem to be a noticeable difference in the average rating of each unique user that we can use to aid our prediction model.

Lets see the number of ratings per user to see if each user have rated a similar number of movies



We can see that the number of ratings from each user does vary quite a bit with some users having over 1000 ratings and some users having only around 20 ratings.

Lets see if the extremes are from users who have given a low number of ratings similar to the movie data

userId	mean_rating	count
1	5	19
7984	5	17
11884	5	18
13027	5	29
13513	5	17

userId	mean_rating	count
13496	0.5	17
48146	0.5	25
49862	0.5	17
62815	0.5	20
63381	0.5	18

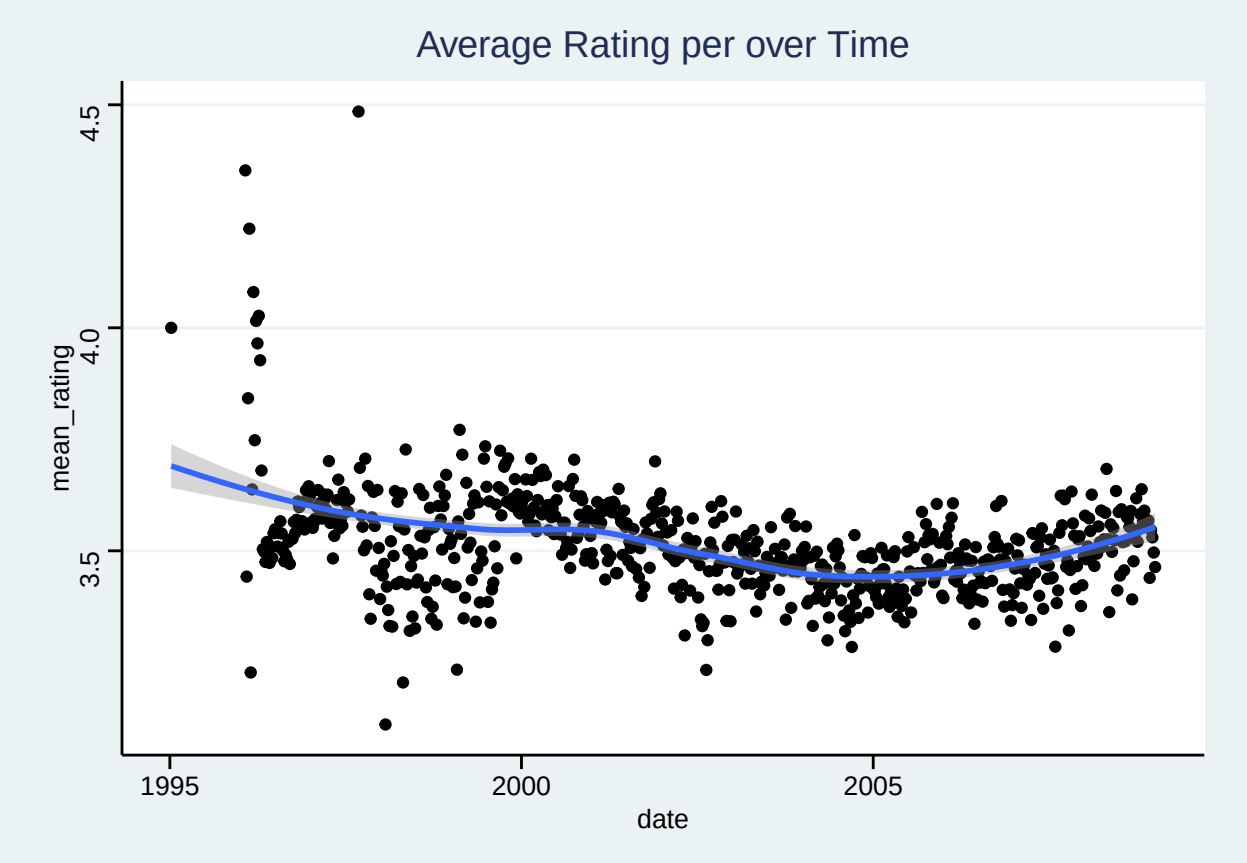
We can see that again the extreme values of the mean ratings per user are from the users who have given the least number of ratings. As there will be the highest amount of variability with the low sample size. So again regularization will have to be used when using user specific data for our prediction model.

Date of review data

Lets find out if the date in which users have rated the movie has an effect on the rating given. As maybe people in general have become more favorable or critical over 14 year range.

Lets observe the trend of the average rating over time rounded to the nearest week

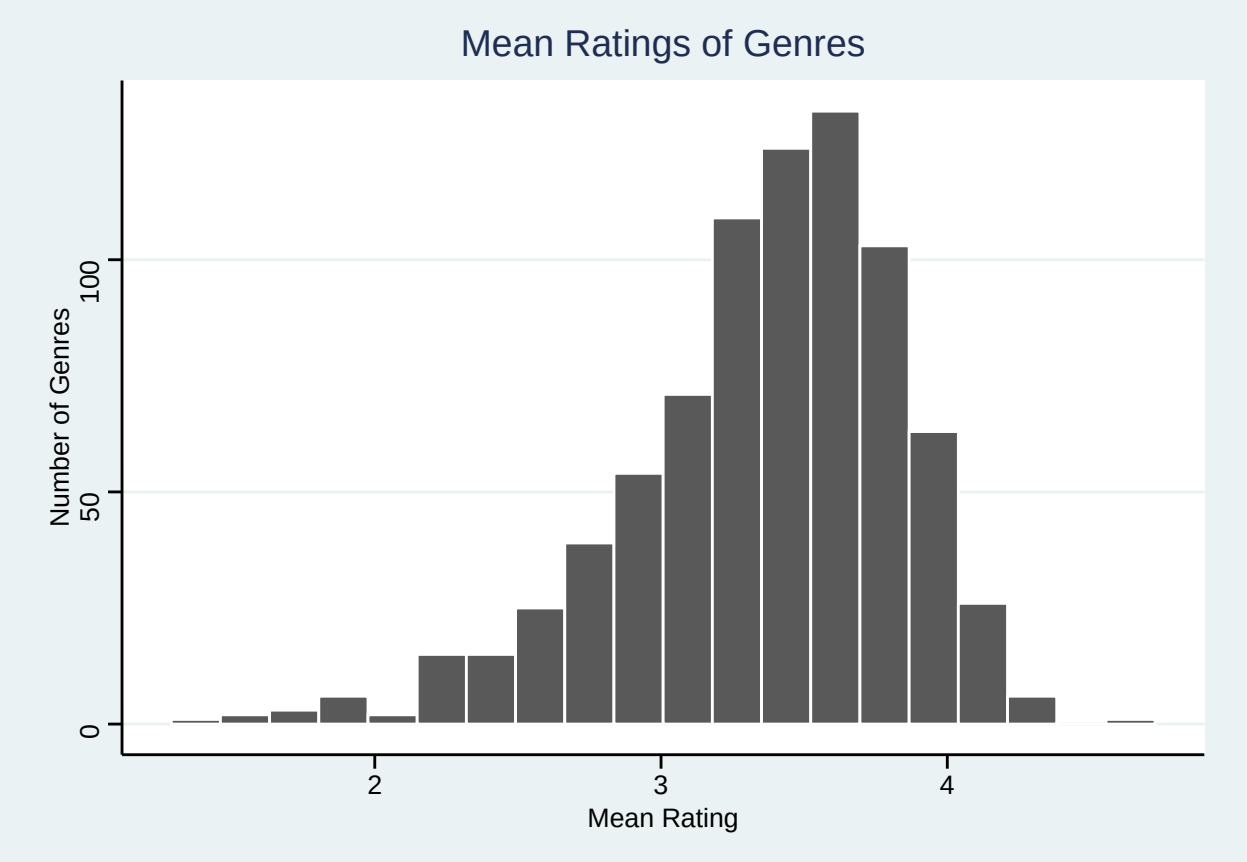
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



We can see a slight change in movie ratings over time but nothing significant enough to include in our model

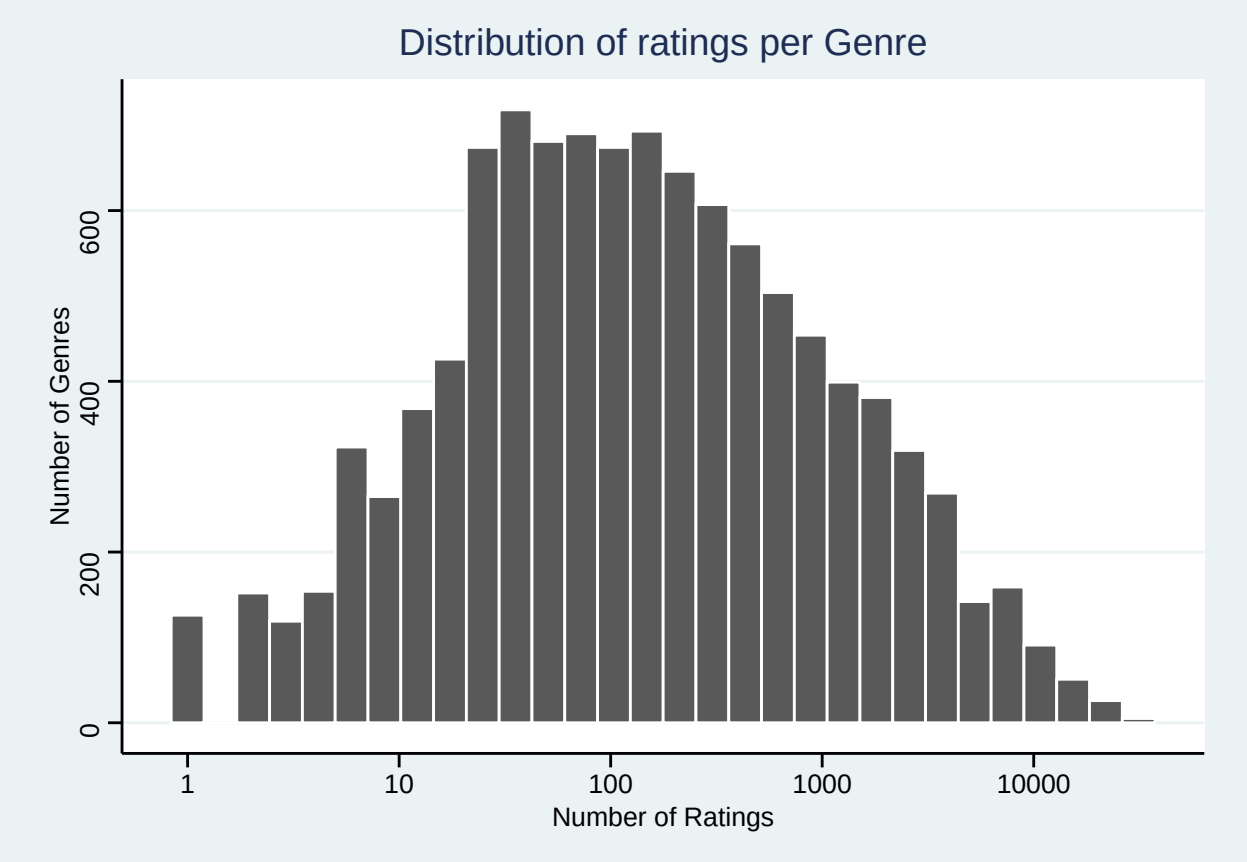
Genres Data

Lets see if 797 different genres have significantly different ratings



We can indeed see that there is quite a spread in the average rating of each genre

Lets see the number of ratings per unique genres to see if each genre combination have a similar number of rating



We can see that there is a vast difference in the number of ratings per genre with some genres having only 1-10 ratings and some having over 10,000 ratings

Lets see if the extremes are from users who have given a low number of ratings similar to the movie data

genres	mean_rating	count
Animation|IMAX|Sci-Fi	4.714286	7
Drama|Film-Noir|Romance	4.304115	2989
Action|Crime|Drama|IMAX	4.297068	2353
Animation|Children|Comedy|Crime	4.275429	7167
Film-Noir|Mystery	4.239479	5988

genres	mean_rating	count
Documentary|Horror	1.449112	619
Action|Animation|Comedy|Horror	1.500000	2
Action|Horror|Mystery|Thriller	1.607034	327
Comedy|Film-Noir|Thriller	1.642857	21
Action|Drama|Horror|Sci-Fi	1.750000	4

We can see that this time there is a mix between genres with few reviews and lots of reviews for the highest and lowest average rating genres. So to severe regularization may not be necessary.

Creating the Prediction Model

First we further split our edx data set into a training set and test set which contain 90% and 10% of the edx data respectively while making sure that all the movieIds and UserIds in the test set are also in the train set.

The model we create is a simple linear model where we add in the effects of different classifiers

Mean

We start by building the simplest possible recommendation system model where we predict all movies with the same rating regardless of the user explaining the differences between ratings with random variation. Defined as follows:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

Where $Y_{u,i}$ is the is the predicted rating for movie i from user u, μ is the mean rating of all movies in the train set with $\epsilon_{u,i}$ representing the independent errors sampled from the same distribution centered at 0 and μ

When predicting all the the unknown ratings of the test set with the equation:

$$\hat{Y}_{u,i} = \hat{\mu}$$

we obtain the following RMSE

model	RMSE
Just mean	1.060054

The RMSE we obtain from this simple model is expectedly quite high

Movie Effect

Next we add in an effect for different movies, as from the analysis we can see that different movies have vastly different ratings. So we augment the previous model by adding in a movie specific effect or movie bias with the term (b_i) which represent the average rating for movie i

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

Additionally however we discovered in the analysis that the movies with the highest and lowest mean ratings were obscure movies with only 1-2 ratings each. So we add in regularization which penalizes large estimates of (b_i) from movies with small sample sizes to constrain the total variability caused. So the calculation for (b_i) is:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

Where (n_i) is the number of ratings made for movie i and λ is the penalty factor applied that reduces the movie bias (b_i) more for movies with small (n_i) than it does for movies with large (n_i). The optimal λ value is chosen via cross validation minimizing the RMSE from the equation:

$$\hat{Y}_{u,i} = \hat{\mu} + \hat{b_i}$$

When inputting the optimal λ obtained from cross validation we get the following RMSE from the model with the added movie effects. We reach a better RMSE of 0.942937

model	RMSE
Just mean	1.060054
Movie bias	0.942937

User Effects

Next we add in an effect in ratings from users to account for more of the remaining variability as we have seen that there is also substantial variability in ratings across different users whereby some users rate movies higher on average and some rate movies lower on average. So a similar method will be used when adding the movie effect whereby:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

Where (b_u) is a user specific effect added

Again we will add in regularization for the (b_u) effect as we saw a similar phenomenon of the extreme user rating averages being from users with the least amount of ratings. So:

$$\hat{b}_u(\lambda) = \frac{1}{\lambda + n_u} \sum_{u=1}^{n_u} (Y_{u,i} - \hat{\mu} - \hat{b_i})$$

After using cross validation again to find the optimal λ we use to equation:

$$\hat{Y}_{u,i} = \hat{\mu} + \hat{b_i} + \hat{b_u}$$

To predict the test set and get a RMSE value of 0.8641744 which is another substantial improvement

model	RMSE
Just mean	1.0600537
Movie bias	0.9429370
Movie + User bias	0.8641744

Genre effect

Next we will add in the effect different genres may to see if that accounts for more of the unexplained variability. As from our analysis we have seen that there is a significant spread in the average ratings of genres. We will add the genre effect(b_g) to our equation:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

Again we will add in regulization again using cross validation to find our lambda values even though from analysis it didn't seem as neccasary as for the previous effects.

$$\hat{b}_g(\lambda) = \frac{1}{\lambda + n_g} \sum_{u=1}^{n_g} (Y_{u,i} - \hat{\mu} - \hat{b_i})$$

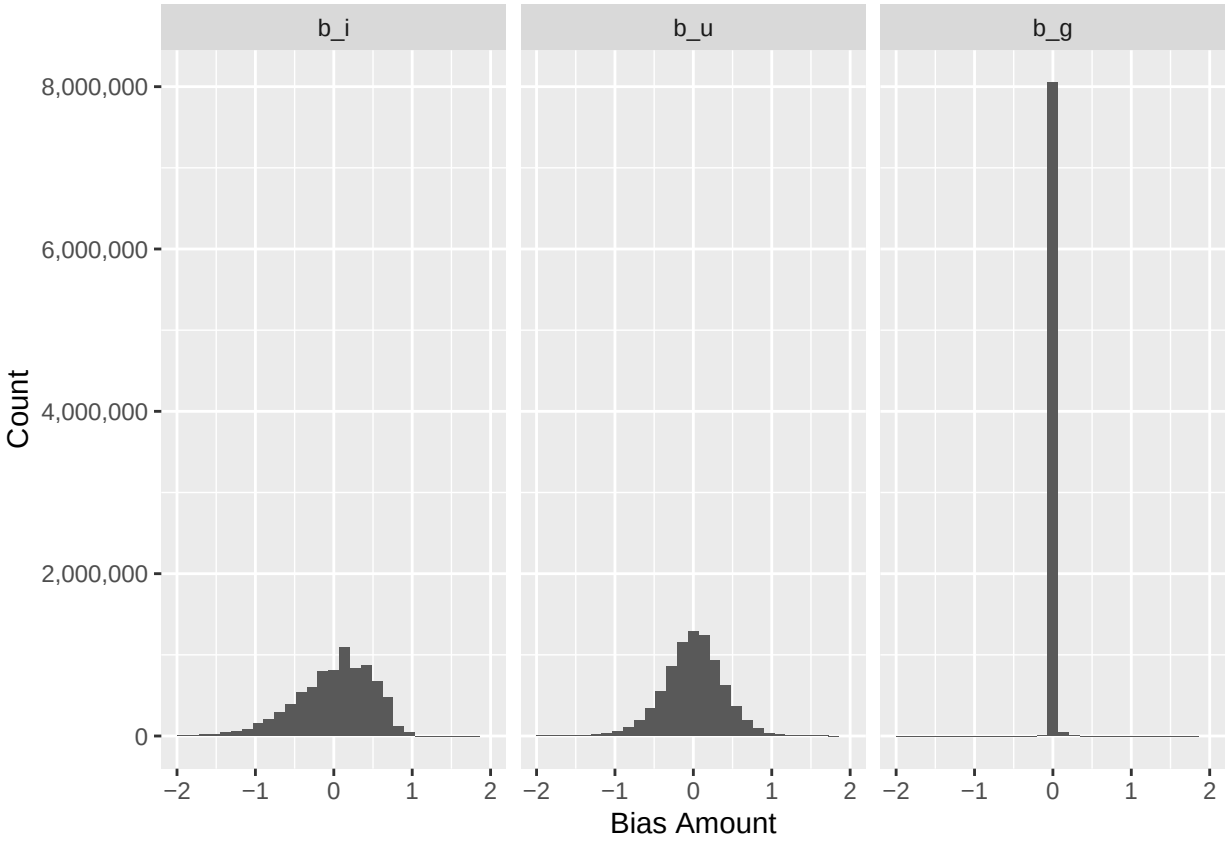
After using cross validation again to find the optimal λ we use to equation:

$$\hat{Y}_{u,i} = \hat{\mu} + \hat{b_i} + \hat{b_u} + \hat{b_g}$$

To predict our test set which results in a new RMSE of .08638564, which is only a very neglible improvement

model	RMSE
Just mean	1.0600537
Movie bias	0.9429370
Movie + User bias	0.8641744
Movie + User + Genres bias	0.8638564

To Visualize why this is we look at the histograms for the effect values (b_i, b_u, b_g)



We can see that the movie bias and user bias have a substantial effect on the variability between ratings while the genre bias has very little effect with values very close 0. This is likely due to the movie and user bias already explaining nearly all of the variability between average ratings between genres and as such we will not include a genre effect as part of our model.

Matrix Facorisation

Matrix factorization is now used to factor in the similar rating patterns that both certain groups of movies and certain groups of users share. As currently the model only pays attention to the fact that both individual movies and individual users have similar rating patterns. The idea is that users who like "The Godfather" more than the current model predicted will also like "The Godfather, Part II" more than the model will predict. Or That users who rate horror movies higher than expected may rate romantic comedy movies lower than expected.

We use matrix factorization by first centering the data to leave only the remaining variation $r_{u,i}$:

$$r_{u,i} = y_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u}$$

```
# Center the data to leave only the remaining variation by removing mu, movie bias and user bias'
train_set <- train_set %>% mutate(residual = rating - mu - b_i - b_u)
```

Then estimating a matrix of residuals where each user gets a row and each column gets a rating. Where $y_{u,i}$is the entry for row u and column i.

The model factorizes the matrix r into a vector P and vector Q, where P is user rating score and Q is movies rating score matrix.

$$r \approx PQ'$$

We use the Recommender system package (recosystem) in R to estimate the residuals matrix

$$r_{u,i}$$

by the product of the two matrices of lower division P_nk and Q_nk

First we create the recosystem object Reco() using the userId, movieId and the residual

```
# Create recosystem
r <- Reco()
train_reco <- data_memory(user_index = train_set$userId, item_index = train_set$movieId, rating = train
```

Then we tune the model using $tune() to find the best tuning parameters. Default tuning parameters are used except to save computation time L1 parameters were ignored as L1 loss minimizes linearly whereas L2 loss minimizes the sum of squares which is what the overall goal is.

```
# Tune the recosystem model
# This part includes multithreading using 6 thread son my PC
# Change "nthread" if you do not have enough
reco_tune <- r$tune(train_reco, opts = list(dim      = c(10L, 20L),
                                            costp_l1 = 0,
                                            costp_l2 = c(0.01, 0.1),
                                            costq_l1 = 0,
                                            costq_l2 = c(0.01, 0.1),
                                            lrate    = c(0.01, 0.1),
                                            niter = 10,
                                            nthreads = 6,
                                            verbose = TRUE))
```

Then we train the model using the best possible tune conducting 50 iterations

```
# Train the recosystem model using optimal tune
r$train(train_reco, opts = c(reco_tune$min, niter = 50))
```

Then export the model using $output()

```
# Export model
reco_mat <- r$output(out_P = out_memory(), out_Q = out_memory())
```

Finally use $predict() to predict the train set and RMSE using the equation:

$$\hat{Y}_{u,i} = \hat{\mu} + \hat{b_i} + \hat{b_u} + \hat{r_u}, i$$

```
# Use recosystem model to predict test set
test_reco <- data_memory(user_index = test_set$userId, item_index = test_set$movieId, index1 = TRUE)
test_residual_pred <- r$predict(test_reco, out_memory())
rmse_mat_fact <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u + test_residual_pred) %>%
  summarize(rmse = RMSE(pred, rating)) %>%
  pull(rmse)

# Add the rmse of the of the model which includes matrix factorization
rmse_results <- rmse_results %>%
  add_row(model = "Matrix Factorisation", RMSE = rmse_mat_fact)

knitr::kable(rmse_results[1:5,])
```

model	RMSE
Just mean	1.0600537
Movie bias	0.9429370
Movie + User bias	0.8641744
Movie + User + Genres bias	0.8638564
Matrix Factorisation	0.7943294

This inclusion of matrix factorization to our model gives us an RMSE of 0.7943294 which is well under our target RMSE of 0.86490

Results

After determining the combined linear model + matrix factorization created from the edx data set is adequate. We apply this model to the validation data set for the final test we we get an RMSE of 0.7946383, which is also well under our target RMSE of 0.86490 acheiving the goal of this project.

```
# Use final model on the validation set
valid_reco <- data_memory(user_index = validation$userId, item_index = validation$movieId, index1 = TRU
valid_residual_pred <- r$predict(valid_reco, out_memory())
rmse_valid <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u + valid_residual_pred) %>%
  summarize(rmse = RMSE(pred, rating)) %>%
  pull(rmse)
```

Conclusion

The model described in this report successfully predicts movie ratings in the 10M MovieLens data set with an RMSE of 0.7946383 which is significantly below the target RMSE of 0.86490. The final model uses a combination of linear modelling of the general quality of the movie, general disposition of the user and then matrix factorization to account for the remaining residuals.

Some limitations to this model are that it is not possible to make predictions for movies and users which have never been accounted for in the training data and would have to be altered to work for new users and movies.

Another limitation is that the matrix factorization aspect of the model is not that interpretable as all the different groupings aren't displayed.

Some aspects that could be looked into further are exploring any extra biases that may be apparent that could be factored into the linear model, such as date the movie came out or time from the first review as people who really like a movie may rush to rate it. Also more parameters could be added to the matrix factorization tuning with more computation power to get a more optimal model.