# MOVIE RECOMMENDATION SYSTEM

HarvardX Data Science Certificate PH125.9X – Capstone

Juan Carlos Lopez

16/10/2024

# Contents

# Chapter 1

# INTRODUCTION

This project is developed within the framework of the HarvardX course PH125.9x - Capstone - Movielens Project and aims to develop a film recommendation system using a dataset.

Generally speaking, a recommendation system is a technology that seeks to anticipate the preference or valuation that a user could assign to an item or product, using specialized algorithms. These systems are applied in a wide variety of fields: Social networks, entertainment, e-commerce, news, among others.

In the specific case of film recommendation systems, it is a technological tool created to suggest films to users, based on several factors.

It mainly serves to:

- Optimisation of the user experience: by offering appropriate suggestions, they increase user satisfaction and involvement.
- Information organisation: they facilitate the management of information overload, selecting what is most relevant for each user and reducing their search time.
- Retention: they keep users interested and active on a platform or service.
- Customisation: tailor the content or products displayed according to the user's individual tastes.
- Exploration: help users discover new items that might interest them, but that they might not have found on their own.
- Catalogue optimisation: enables streaming services to highlight their content more efficiently.
- For companies, information from these recommender systems can be very useful when deciding on the type of content to produce, based on historical data provided by their customers. A representative example of this application was given when Netflix acknowledged that it had used this type of information in the decision process to produce the series 'House of Cards'. (although the Movielens dataset only includes information on films, in practice it also collects information on TV series).

## 1.1 Objective

The goal of this project is to create a movie recommendation system based on the MovieLens 10M dataset.

The model to be developed must achieve a Root Mean Square Error (RMSE) < 0.8649.

## 1.2 Dataset Description

[MovieLens] (https://movielens.org/) is an online platform developed by GroupLens Research (https://grouplens.org/) at the University of Minnesota, which provides a widely-used database for recommendation system research, as well as personalized movie recommendations to its online users.

They have developed different versions of their dataset, with MovieLens 10M (https://grouplens.org/datasets/movielens/10m/) being the one used in this project. This dataset was published in 2009 and contains approximately 10 million ratings, made by about 72,000 users for 10,000 movies.

## 1.3 Project Phases

The steps to follow will be, in summary:

- Download the MovieLens 10M dataset, which contains information about movie ratings given by a large number of users.
- Split the original dataset into two subsets, using the code provided by the course instructors, in order to use one for training the algorithm and the other for validating the final model.
- Analyze the structure and characteristics of the data.
- Generate the necessary models until achieving or surpassing the course's target of a Root Mean Square Error (RMSE) < 0.8649.
- Present results and conclusions.

More detailed explanations will be provided throughout this report.

Note: Due to the data wrangling performed on the original dataset, which has resulted in a considerable increase in its size, I recommend running the code on platforms like Posit Cloud or Google Colaboratory to avoid long execution times.

# Chapter 2

# DATA: PREPARATION AND ANALYSIS

Before starting the model construction, the data needs to be downloaded, prepared, and analyzed

## 2.1 Data Download and Preparation

With the following code, provided in the course, we will download the data and split the mentioned dataset into two different sets, called 'edx' (90% of the data) and 'final_holdout_test' (remaining 10%). The first one will be used to develop the model, and the second one for the final validation.

```
############################################################
# Create edx and final_holdout_test sets
############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse))
install.packages("tidyverse"
, repos = "http://cran.us.r-project.org")
if(!require(caret))
install.packages("caret"
, repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip
```

```r
options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
download.file(paste0("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)
ratings <- as.data.frame(str_split(read_lines(ratings_file),
                                   fixed("::"),
                                   simplify = TRUE),
                         stringsAsFactors = FALSE)
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")

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
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 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 test
# set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

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

Additionally, we will use the following packages:

```
if(!require(data.table))
 install.packages("data.table"
, repos = "http://cran.us.r-project.org")
if(!require(knitr))
 install.packages("knitr"
, repos = "http://cran.us.r-project.org")
if (!require(scales))
 install.packages("scales"
, repos = "http://cran.us.r-project.org")
if (!require(ggthemes))
 install.packages("ggthemes"
, repos = "http://cran.us.r-project.org")
if (!require(recosystem))
 install.packages("recosystem"
, repos = "http://cran.us.r-project.org")
if(!require(formatR))
 install.packages("formatR"
, repos = "http://cran.us.r-project.org")

library(data.table)
library(knitr)
library(scales)
library(ggthemes)
library(recosystem)
library(formatR)
```

## 2.2 Preliminary Exploration

We will start by analyzing the data structure, its characteristics, and the relationship between the variables:

Displaying the first 5 rows:

**First 5 Rows of the edx Dataset**

| 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 |

Displaying the number of rows and columns

**Dimensions Table**

| Description | Value |
|---|---|
| Number of Rows | 9000055 |
| Number of Columns | 6 |

Checking for missing values:

**Missing Values in edx Dataset**

| Column | Missing_Values |
|---|---|
| userId | 0 |
| movieId | 0 |
| rating | 0 |
| timestamp | 0 |
| title | 0 |
| genres | 0 |

Range:

| Unique Values in edx Dataset | |
|---|---|
| **Column** | **Unique_Values** |
| userId | 69878 |
| movieId | 10677 |
| rating | 10 |
| timestamp | 6519590 |
| title | 10676 |
| genres | 797 |

Variable type:

| Data Types in edx Dataset | |
|---|---|
| **Column** | **Data_Type** |
| userId | integer |
| movieId | integer |
| rating | numeric |
| timestamp | integer |
| title | character |
| genres | character |

Description of each variable:

- userId: contains the identification of each user, with 69,878 distinct values.
- movieId: numeric identifier associated with each movie, with 10,677 distinct values.
- rating: users' rating for each movie, in multiples of 0.5, ranging from 0.5 to 5.
- timestamp: the date when the rating was made, expressed in seconds since midnight on January 1st, 1970.
- title: the name of the movie, including the release year.
- genres: classification based on the genre of each movie, with 797 distinct values.

Comments: when analyzing the table above, we observe:

- The timestamp column is in a difficult-to-read format. Additionally, we can convert it into a rating year to study its possible effect.

- Genres: although this may increase the size of the dataset, it is advisable to convert multiple classifications into individual ones, with the same goal as the previous point.
- Title: besides the movie title, it provides information about the release year, considering this information valuable enough to separate it.

## 2.3 Data wrangling

### 2.3.1 timestamp

We convert the current format into one that shows the year in which each movie was rated by the user. Therefore, we remove this variable from the dataset and create a new one called Rating_Year.

**First 5 Rows of the edx Dataset (After Timestamp Conversion)**

| userId | movieId | rating | title | genres | Rating_Year |
|---|---|---|---|---|---|
| 1 | 122 | 5 | Boomerang (1992) | Comedy\|Romance | 1996 |
| 1 | 185 | 5 | Net, The (1995) | Action\|Crime\|Thriller | 1996 |
| 1 | 292 | 5 | Outbreak (1995) | Action\|Drama\|Sci-Fi\|Thriller | 1996 |
| 1 | 316 | 5 | Stargate (1994) | Action\|Adventure\|Sci-Fi | 1996 |
| 1 | 329 | 5 | Star Trek: Generations (1994) | Action\|Adventure\|Drama\|Sci-Fi | 1996 |

### 2.3.2 title

It is about extracting the release year of the movie, moving it to a new column called Release_Year, and removing the year from the title column.

**First 5 Rows of the edx Dataset (After Release Year Extraction)**

| userId | movieId | rating | title | genres | Rating_Year | Release_Year |
|---|---|---|---|---|---|---|
| 1 | 122 | 5 | Boomerang | Comedy\|Romance | 1996 | 1992 |
| 1 | 185 | 5 | Net, The | Action\|Crime\|Thriller | 1996 | 1995 |
| 1 | 292 | 5 | Outbreak | Action\|Drama\|Sci-Fi\|Thriller | 1996 | 1995 |
| 1 | 316 | 5 | Stargate | Action\|Adventure\|Sci-Fi | 1996 | 1994 |
| 1 | 329 | 5 | Star Trek: Generations | Action\|Adventure\|Drama\|Sci-Fi | 1996 | 1994 |

### 2.3.3 genres

In this section, the current content of the genres column (one or more genre classifications for each movie) is broken down into individual classifications.

This action will result in a substantial increase in the amount of data, corresponding slowdown in code execution, and the possibility of obtaining incorrect results in our calculations due to data multiplication. Therefore, the functions distinct() and sometimes group_by() will need to be used.

Despite these inconveniences, we consider the information we will obtain to be valuable

**First 5 Rows of the edx Dataset (After Genres Separation)**

| userId | movieId | rating | title | genres | Rating_Year | Release_Year |
|---|---|---|---|---|---|---|
| 1 | 122 | 5 | Boomerang | Comedy | 1996 | 1992 |
| 1 | 122 | 5 | Boomerang | Romance | 1996 | 1992 |
| 1 | 185 | 5 | Net, The | Action | 1996 | 1995 |
| 1 | 185 | 5 | Net, The | Crime | 1996 | 1995 |
| 1 | 185 | 5 | Net, The | Thriller | 1996 | 1995 |

.- The modifications made to the edx dataset have caused its dimensions to change:

**Dimensions of the edx Dataset**

| Description | Value |
|---|---|
| Number of Rows | 23371423 |
| Number of Columns | 7 |

.- Obviously, we also need to apply these three processes to the dataset that we will use to test the final model, `final_holdout_test`.

**First 5 Rows of the final_holdout_test Dataset**

| userId | movieId | rating | title | genres | Rating_Year | Release_Year |
|---|---|---|---|---|---|---|
| 1 | 231 | 5 | Dumb & Dumber | Comedy | 1996 | 1994 |
| 1 | 480 | 5 | Jurassic Park | Action | 1996 | 1993 |
| 1 | 480 | 5 | Jurassic Park | Adventure | 1996 | 1993 |
| 1 | 480 | 5 | Jurassic Park | Sci-Fi | 1996 | 1993 |
| 1 | 480 | 5 | Jurassic Park | Thriller | 1996 | 1993 |

## 2.4 Analysis of the variables

Now that we have the dataset prepared, we will proceed with its analysis using various techniques, including visualization, to assess the suitability of including each variable in our model.

### 2.4.1 rating

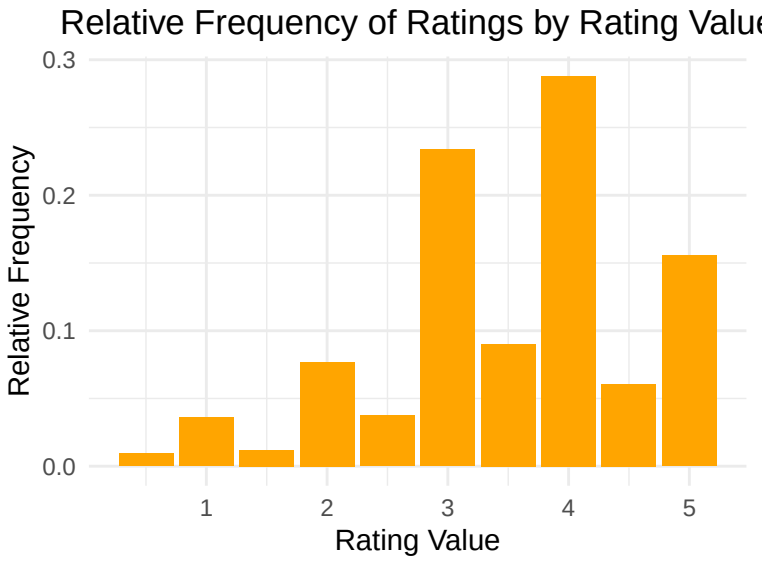
As already mentioned, users can rate movies with a score ranging from 0.5 to 5, in multiples of 0.5

| Rating_1 | Rating_2 | Rating_3 | Rating_4 | Rating_5 | Rating_6 | Rating_7 | Rating_8 | Rating_9 | Rating_10 |
|---|---|---|---|---|---|---|---|---|---|
| 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 |

.- Calculating the mean: As we have already mentioned, we need to remove the duplicate rows that arise after splitting genres in order to calculate the mean

| Mean Rating |
|---|
| 3.512465 |

.- For knowing the relative frequency:



Comments:

- The chart shows the proportion of each rating value relative to the total ratings in the dataset.
- The most frequently used rating is 4 (approximately 30%).
- Ratings below 3 have much lower frequencies, with a minimum at 0.5 and a slight increase at 2.0. This suggests that very low ratings are rare.
- Users tend to give positive ratings to movies, as most ratings are concentrated around 3, 4, and 5. This could indicate a tendency toward satisfaction or a positive bias in the ratings.

.- There is a clear tendency to use rounded values:

| rating_type | count |
|---|---|
| Non-integer Ratings | 4895871 |
| Integer Ratings | 18475552 |

.- The table shows two categories of ratings:

- Integer Ratings: Whole number ratings such as 1, 2, 3, 4, and 5.
- Non-integer Ratings: Decimal ratings such as 0.5, 1.5, 2.5, 3.5, and 4.5.

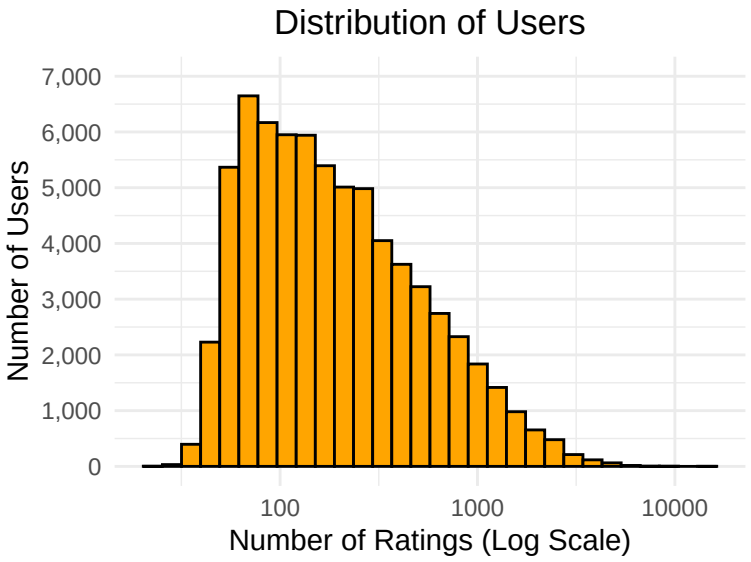
.- Integer ratings are much more frequent than non-integer ratings. This could indicate that users tend to rate using whole numbers instead of decimals. In some recommendation systems, users might prefer using whole numbers because they seem more straightforward and less ambiguous. Another explanation could be that the system encourages the use of whole numbers if the interface defaults to integer options or if it is simply more convenient for users to select a whole number.

### 2.4.2 userId

.- As previously mentioned, the dataset contains 69,878 distinct users.

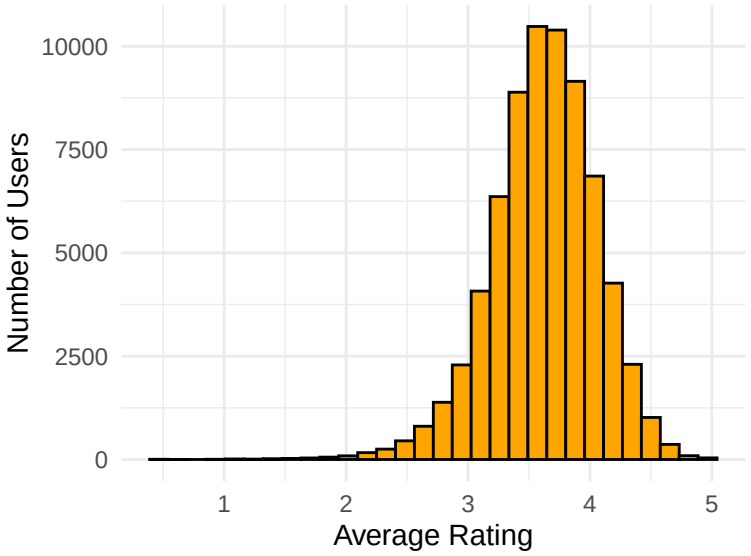
| Users |
|---|
| 69878 |

.- We are interested in understanding how the ratings given by users are distributed:




- The distribution is right-skewed, which implies that most users are less active (few ratings), while only a few are very active (many ratings).
- The distribution has a peak in the range of 50 to 100 ratings per user, indicating that most users have given between 50 and 100 ratings.
- The number of users decreases as the number of ratings increases, showing that there are few users who rate many movies.

.- It is also interesting to know the average ratings:

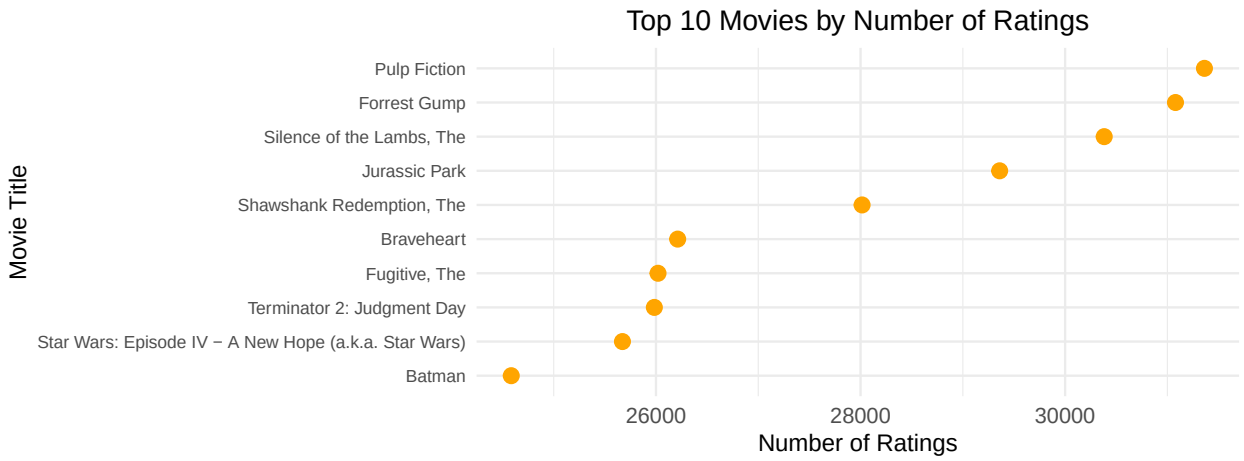


- The distribution is approximately symmetric, with a slight rightward skew. This type of distribution suggests that most users tend to give ratings slightly above the average.
- There are very few users with extremely low or extremely high average ratings. This may indicate that most users have a balanced distribution of their ratings.

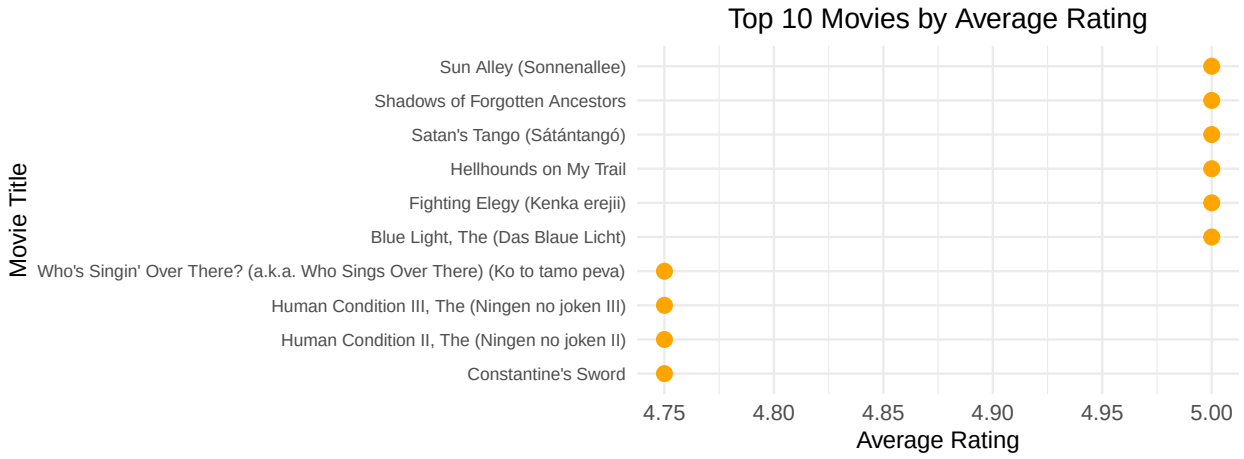
### 2.4.3 title

As previously mentioned, there are 10,677 movies in the dataset.

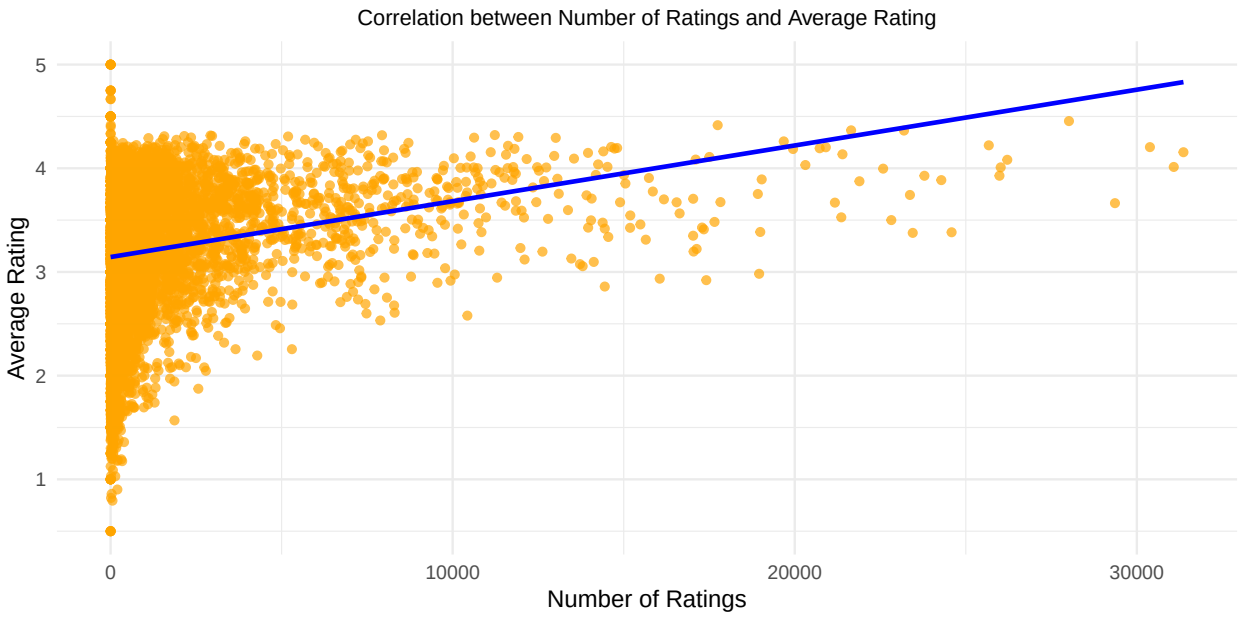
.- Let's analyze which ones have received the most ratings:




.- Now let's look at the movies that have received the highest ratings:



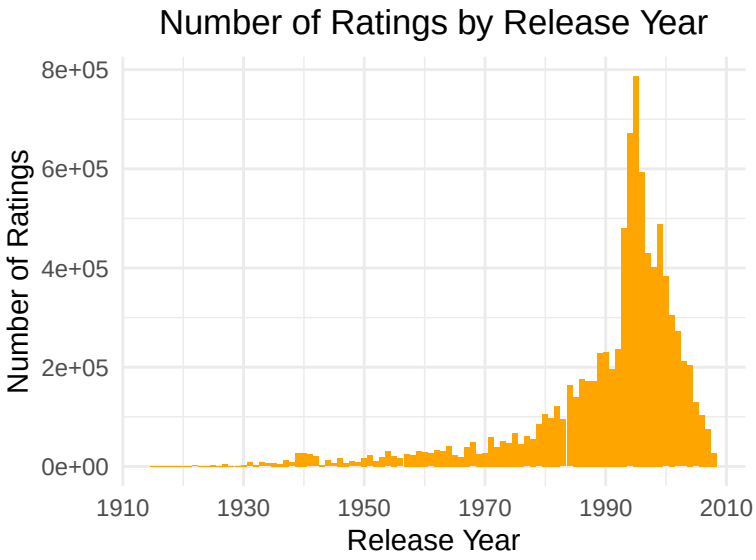

.- In other words, the movies that have received the best ratings are not the ones that have received the highest number of ratings. Let's examine this by studying the correlation between the number of ratings and the average rating:




- The chart shows the relationship between the Number of Ratings and the Average Rating for each movie in the dataset.
- There is a high concentration of points on the left side of the chart, particularly in low values of Number of Ratings (fewer than 500). This suggests that many movies have received few ratings.
- The trend line has a positive slope, indicating a weak positive correlation between the number of ratings and the average rating. In general, as the number of ratings increases, the average rating tends to be slightly higher.
- However, despite the overall positive trend, there is significant dispersion. Some movies with few ratings have very low or very high ratings, while movies with many ratings have ratings more concentrated around the average (between 3 and 4).

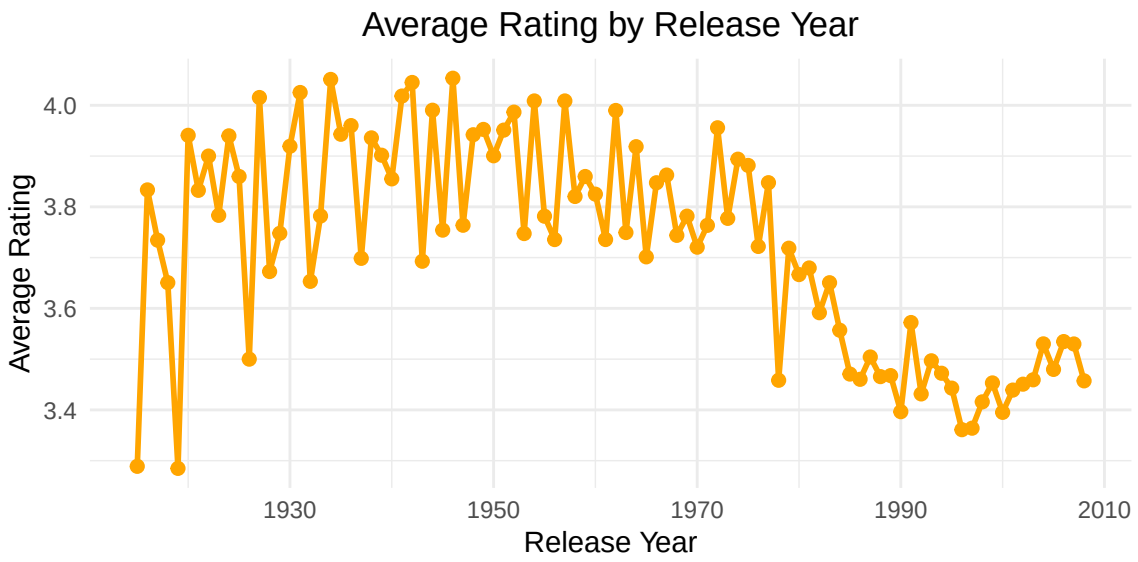
### 2.4.4 Release_Year

Let's see if there is any relationship between the release year and the number of ratings received:




- There is a gradual increase in the number of ratings from 1920 to 1980. This may be due to the growing availability and popularity of movies released in those years.
- A significant peak in the number of ratings is observed for movies released during the 1990s, reaching the highest point around 1995. After 1995, there is a noticeable decline in the number of ratings for movies released in later years, although a considerable amount of ratings continues up until 2005.
- Possible explanations for these facts could be:
- Movies released in the 90s may have been more popular and more available on the platform where the data was collected, which explains the large number of ratings.
- The large number of ratings for movies prior to the 80s and 70s may be due to classic or cult films that remain popular among users.

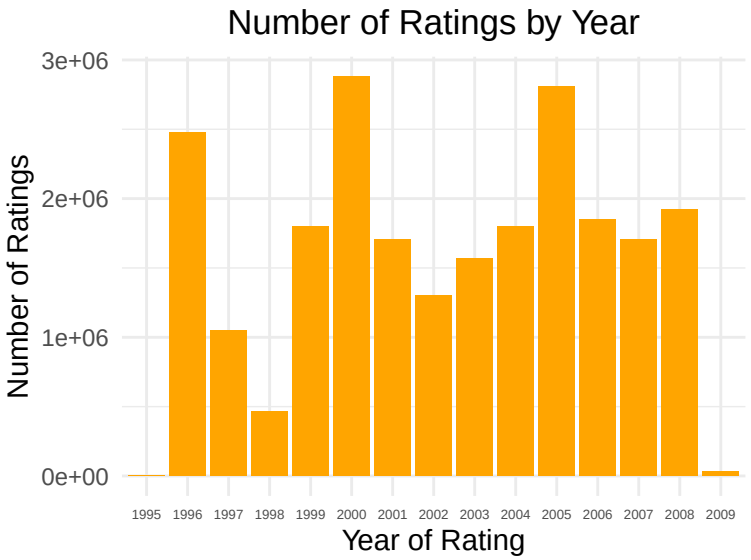
.- Now let's see if there is a relationship between the release year and the rating received:




- A general downward trend is observed in the average rating over time, especially from the mid-20th century to the early 2000s. This indicates that movie ratings tend to be lower in more recent years.
- In the early decades of the 20th century, there is high variability in average ratings, with significant peaks and valleys. This could be due to the smaller number of movies and ratings in those years, making the averages more volatile.

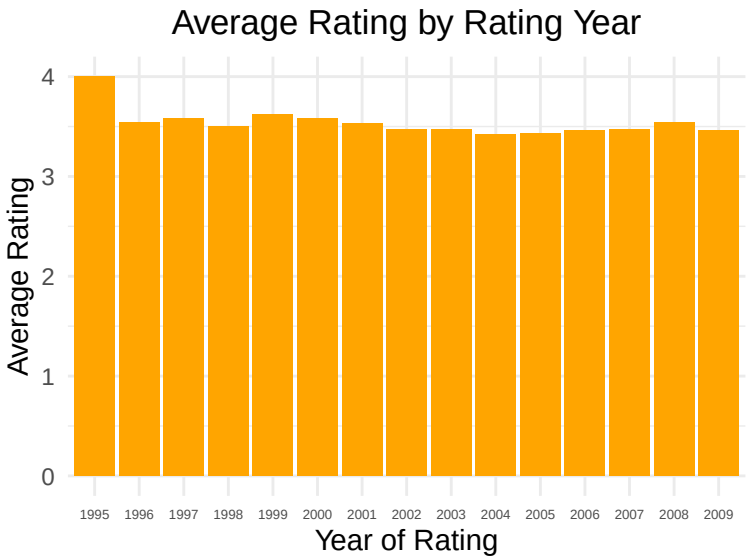
### 2.4.5 Rating_Year

It might be interesting to know if there is a relationship between the number of ratings and the year in which the ratings were produced, in order to detect how long it took for this practice to become widespread among users, identify trends, etc.




.- The variability in the number of ratings could reflect changes in user behavior, such as peaks of activity in certain years followed by stabilization or decline.

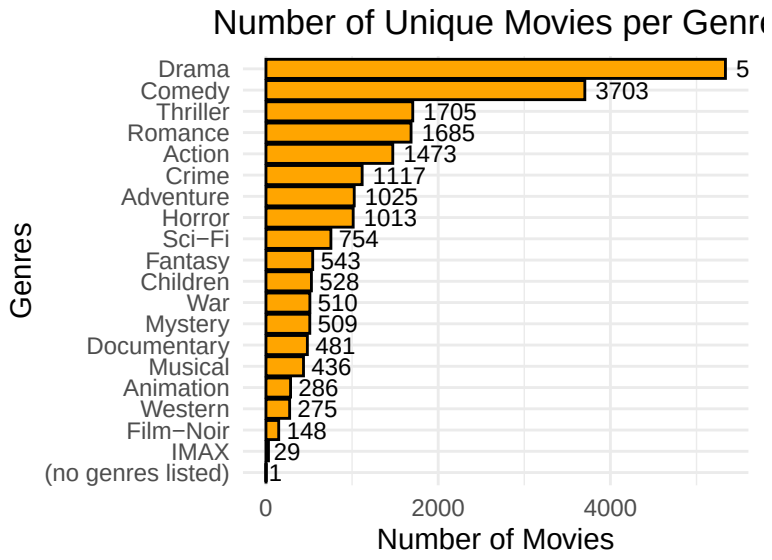
.- To know the average rating depending on the rating year:



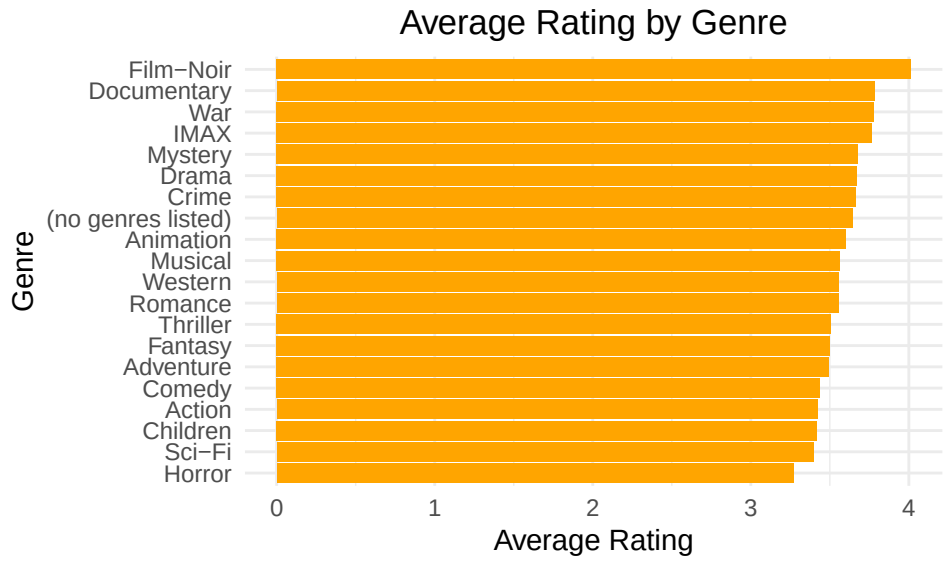

.- The consistency in average ratings from 1996 to 2009 indicates that, regardless of the number of ratings given, the users' overall perception of movies did not change drastically in general terms

### 2.4.6 genres

Initially, the dataset contained 797 distinct genre combinations. After the transformation, we are left with 20, distributed as follows:



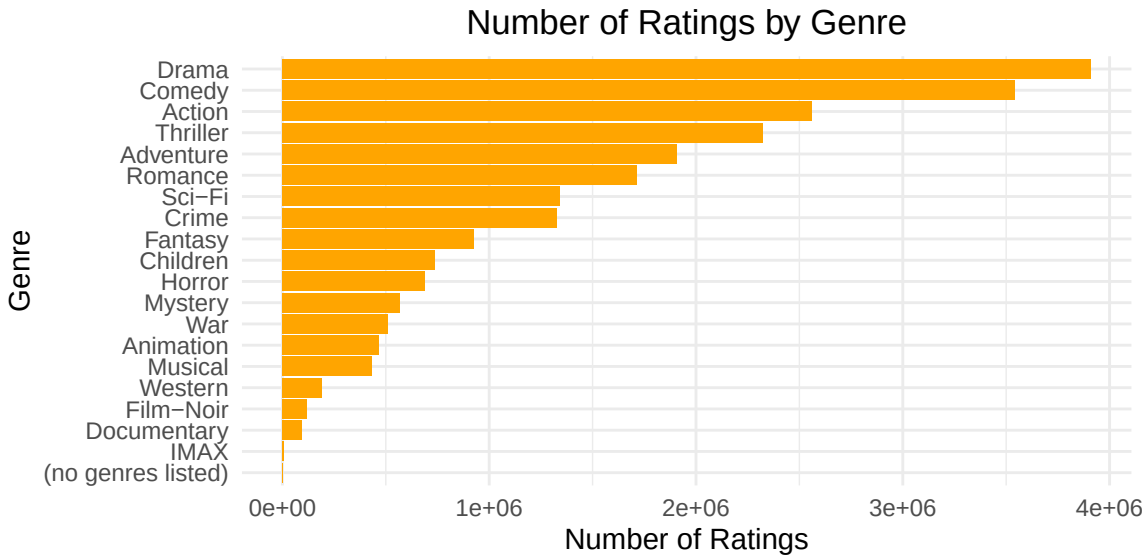

.- To know the average rating each genre receives:




- Genres like Film-Noir and Documentary tend to have a more specific and less mainstream audience. This could lead to higher ratings, as people who consume these genres tend to be enthusiasts and give more positive reviews.

- Genres like Comedy, Action, and Children, although popular, usually have a wide range in terms of perceived quality, which can lead to a dispersion of ratings and, consequently, a lower average.

.- To know how many ratings each genre has received:




- Drama, Comedy, and Action are the genres that have received the highest number of ratings, with over 3 million ratings each. This suggests that they are widely popular genres consumed by a large number of users. Thriller and Adventure also have a high number of ratings, indicating considerable interest in these genres.

- Genres like Drama, Comedy, and Action tend to be more accessible and popular, attracting a broader audience and therefore receiving more ratings. Genres like Documentary and Film-Noir tend to attract a more specific, niche audience, which is reflected in a lower number of ratings.

# Chapter 3

# MODELING

Before starting to develop the successive models that will lead us to achieve the desired goal, we need to split the edx dataset into two subsets to avoid overfitting our model.

Using part of the code provided by the course instructors, we will split edx into two parts: the first, called edx2, will contain 90% of the data and will be used for the development of successive models; the second, containing the remaining 10% of the data, will be called testing, and we will use it to measure the accuracy of the developed models.

Let's not forget that the validation of the final model must be done using the final_holdout_test dataset.

To check that the partition has been done correctly:

.- First 5 rows of edx2:

First 5 Rows of the edx2 Dataset

| userId | movieId | rating | title | genres | Rating_Year | Release_Year |
|---|---|---|---|---|---|---|
| 1 | 122 | 5 | Boomerang | Romance | 1996 | 1992 |
| 1 | 185 | 5 | Net, The | Action | 1996 | 1995 |
| 1 | 185 | 5 | Net, The | Crime | 1996 | 1995 |
| 1 | 292 | 5 | Outbreak | Action | 1996 | 1995 |
| 1 | 292 | 5 | Outbreak | Drama | 1996 | 1995 |

.- Now, the same with testing:

First 5 Rows of the testing Dataset

| userId | movieId | rating | title | genres | Rating_Year | Release_Year |
|---|---|---|---|---|---|---|
| 1 | 122 | 5 | Boomerang | Comedy | 1996 | 1992 |
| 1 | 185 | 5 | Net, The | Thriller | 1996 | 1995 |
| 1 | 329 | 5 | Star Trek: Generations | Sci-Fi | 1996 | 1994 |
| 1 | 466 | 5 | Hot Shots! Part Deux | Action | 1996 | 1993 |
| 1 | 539 | 5 | Sleepless in Seattle | Comedy | 1996 | 1993 |

.- About dimensions:

Dataset Dimensions

| Dataset | Rows | Columns |
|---|---|---|
| edx original | 9000055 | 6 |
| edx modified | 23371423 | 7 |
| edx2 | 21034287 | 7 |
| testing | 2337136 | 7 |

The accuracy of the successive models that will be developed will be measured using the Residual Mean Squared Error:

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

where:

$n$: represents the total number of predictions being evaluated.

$\hat{y}_{u,i}$: the rating predicted by the model for user $u$ and movie $i$.

$y_{u,i}$: the actual rating that user $u$ gave to movie $i$.

- The RMSE measures the average error between the ratings a model predicts and the actual ratings given by users. In other words, it indicates how close the predictions are to the actual ratings.
- A low RMSE would indicate that the predicted ratings are close to the actual ratings. This means that the recommendation system is making good predictions, while a high RMSE would indicate that the model's predictions are far from the actual ratings, suggesting that the model needs improvement as it is generating inaccurate predictions

## 3.1 Average movie rating model

In this first model, the prediction of the rating $\hat{y}_{ui}$ for any movie i and any user u is based exclusively on the global mean μ of all ratings in the dataset. However, we acknowledge that there are individual rating differences that cannot be explained solely by the mean, so we introduce the random term $\epsilon_{u,i}$ to capture inherent noise in the predictions

$$\hat{y}_{u,i} = \mu + \varepsilon_{u,i}$$

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |

## 3.2 Movie effect model

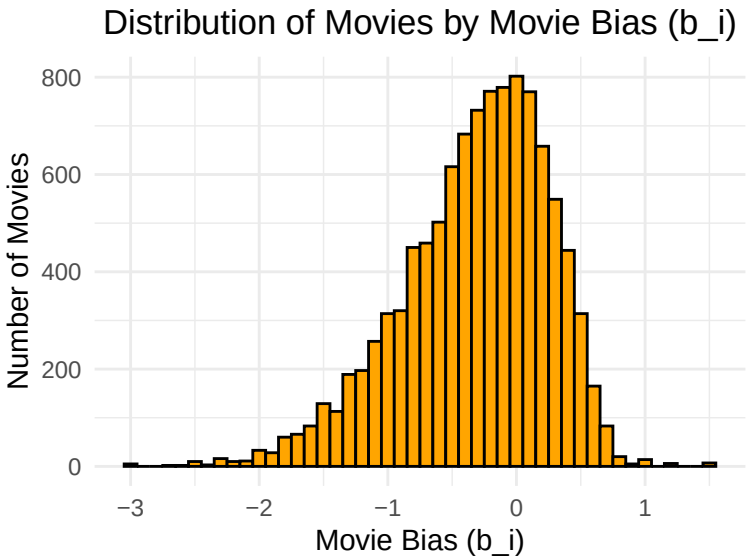
As expected, the previous model yields a very high RMSE, indicating that it is not sufficiently accurate in its predictions. To improve the model's performance, we propose adding the effect caused by the movie variable

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

where b_i is the bias of movie i, which measures how much the average rating of movie i deviates from the global mean μ.

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.94100 |

.- Graphically analyzing the movie bias:



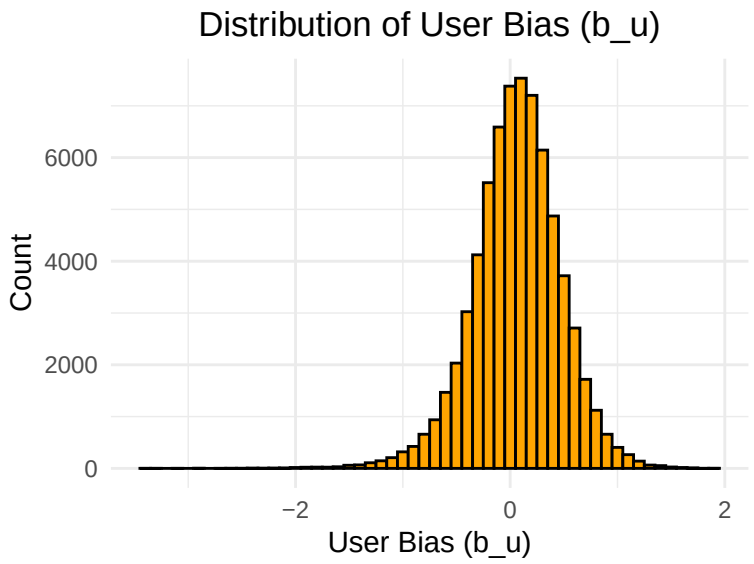

## 3.3 Movie and User effects model

The previous model, which includes the global mean μ and the movie bias b_i, improves prediction accuracy by considering that different movies have different average ratings. However, we still do not capture the fact that different users have different rating patterns. For example, some users tend to give higher ratings than others, regardless of the movie, while others may be more critical and generally give lower ratings. To reflect this, we add a new term to the model, b_u, which captures the user-specific bias.

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

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.94100 |
| Movie+User_Effect Model | 0.85561 |

.- To generate the bias plot:




.- The concentration of most values around 0 indicates that the majority of users do not deviate much from the adjusted mean. This suggests relatively uniform rating behavior

## 3.4 Movie, User & Release Year effects model

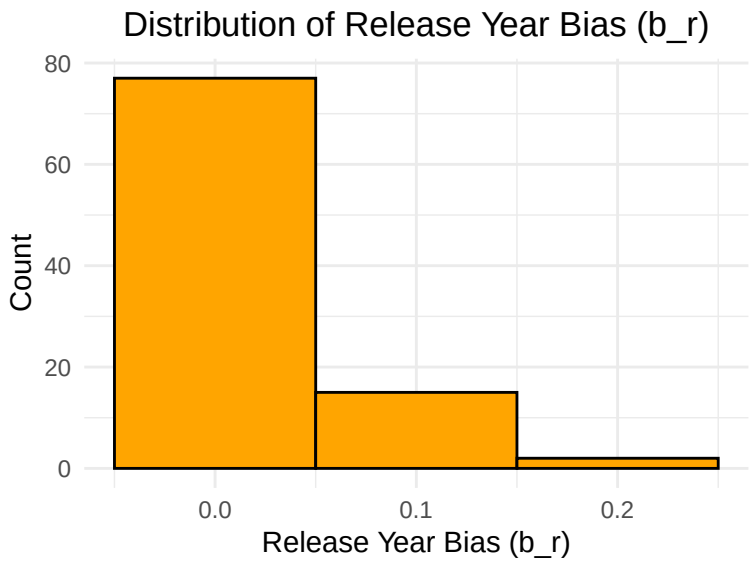
After introducing the effect of users and movies into the model, we significantly improved the system's ability to make personalized predictions. However, we are still not accounting for the fact that ratings may vary depending on the age of the movies. Films released in different eras may receive different ratings due to factors such as changes in cinematic trends, audience preferences over time, or technical quality.

To account for this phenomenon, we add a new term to the model, b_r, which adjusts the predictions based on the year the movie was released.

$$\hat{y}_{u,i} = \mu + b_i + b_u + b_r + \epsilon_{u,i}$$

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.94100 |
| Movie+User_Effect Model | 0.85561 |
| Movie+User+Release_Year Model | 0.85529 |

.- Analyzing it visually:




.- The concentration of most values around 0 suggests that the release year is not a determining factor for the deviation from the average rating once the movie and user biases have been adjusted for.

## 3.5 Movie, User, Release Year & Genres effects model

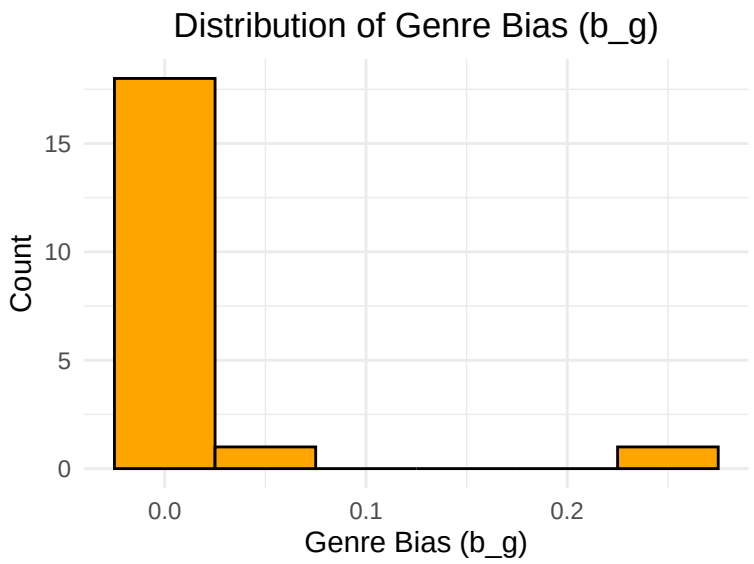
After incorporating the effects of users, movies, and release year into the model, we are still missing an important dimension that may strongly influence ratings: the movie genre. The genre of a movie (such as action, drama, comedy, etc.) can have a significant impact on how users rate the movie, as different genres tend to attract different types of audiences, and some genres tend to be rated higher in general.

To incorporate this phenomenon, we add a new term, b_g, which represents the effect of the movie genre.

$$\hat{y}_{u,i} = \mu + b_i + b_u + b_r + b_g + \epsilon_{u,i}$$

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.94100 |
| Movie+User_Effect Model | 0.85561 |
| Movie+User+Release_Year Model | 0.85529 |
| Movie+User+Release_Year+Genre Model | 0.85498 |

.- To analyze the bias graphically:



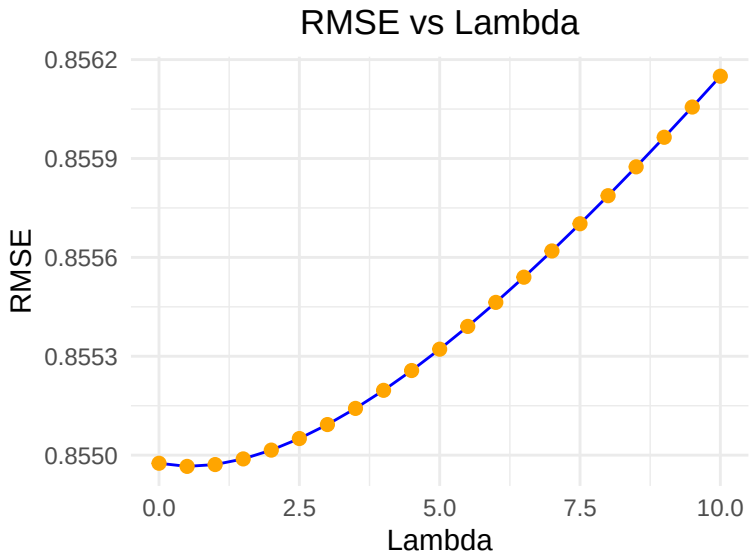

- The concentration of most values around 0 suggests that, once the effects of movie, user, and release year are adjusted, the perception of genres in terms of rating is quite uniform.
- There are no genres with a considerable negative bias, indicating that no specific genre is rated significantly worse than the adjusted average.
- The distribution suggests that genre, as an individual factor, has a limited impact on the deviation from the average rating when the other factors are considered.

## 3.6 Regularization of the model with the effects of movie, user, Release_Year & genres

Regularization is a fundamental concept in the field of machine learning and statistics, especially when building recommendation and regression models. Its main purpose is to prevent overfitting, that is, when a model fits the training data too well and fails to generalize to new data.

In general terms, regularization is a technique that penalizes the complexity of the model by adding a penalty term to the cost or error function. This helps to prevent the model from fitting the training data too closely, thereby improving its generalization capability.

.- Visualizing the relationship between RMSE and lambda:




.- To find the optimal lambda:

```
## The optimal value of lambda is: 0.5
```

```
## The minimum associated RMSE is: 0.8549659
```

.- And now we apply the optimal lambda to the model:

```
## RMSE of final model with optimal lambda is: 0.8549659
```

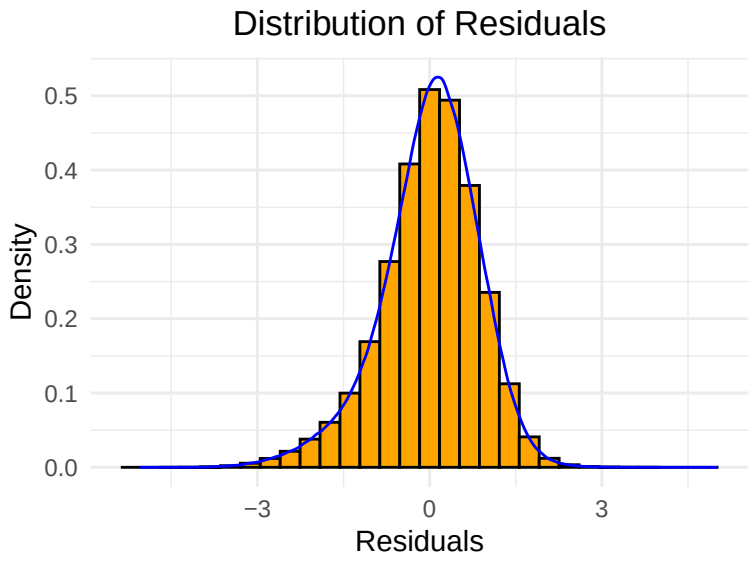
.- Comparing the obtained RMSE with those from previous models:

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.941 |
| Movie+User_Effect Model | 0.85561 |
| Movie+User_Release_Year Model | 0.85529 |
| Movie+User_Release_Year+Genre Model | 0.85498 |
| Regularized Model | 0.85497 |

It seems that the RMSE barely improves after applying regularization. This type of result can occur under certain circumstances, and here are some possible reasons:

- Already generalized model: It's possible that the original model is already well generalized, meaning it's not overfitting.
- Regularization reduces the values of the effects ($b_i$, $b_u$, , etc.) towards zero. If these effects are already close to zero or if the data is very consistent, regularization will not have a significant impact.

Residuals, that is, the difference between the predicted and actual values, can tell a lot about the model's performance. A residual plot should show a distribution close to zero, without any obvious patterns




.- The model is doing a good job overall, due to the high concentration of residuals around 0.

## 3.7 Matrix factorization

In an attempt to improve our model, we will use matrix factorization, which consists of decomposing a large matrix into the product of two (or more) smaller matrices. It is a very useful tool in recommendation models because it allows predictions on missing data, such as ratings that users have not yet given to movies. In general terms, it is a way to reduce the dimensionality of the problem and capture latent patterns in the data.

$$A \approx P \times Q^T$$

where:

$A$ is the original matrix: the rows represent the users, the columns represent the movies, and the values in the cells are the ratings that users have given to the movies.

$P$ is the matrix of user factors. Each row represents a user, and each column represents a latent factor (a hidden characteristic that affects their ratings).

$Q^T$ is the matrix of movie factors. Each row of $Q$ represents a movie, and each column represents how this movie relates to the same latent factors as $P$ , and T is simply a notation to indicate that we are using the transposition of the matrix Q so that the dimensions are compatible for multiplication with $P$ .

It is important to mention that, due to the lack of processing capacity, I had to reduce the number of dimensions, the number of iterations, and simplify the regularization parameters and learning rate, so the result obtained is not optimal.

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.941 |
| Movie+User_Effect Model | 0.85561 |
| Movie+User_Release_Year Model | 0.85529 |
| Movie+User_Release_Year+Genre Model | 0.85498 |
| Regularized Model | 0.85497 |
| Matrix Factorization Model | 0.79894 |

## 3.8 Final validation

Once we have reached our best model using the edx partitions (edx2 and testing), as described at the beginning of our report, we need to validate the final model with the datasets into which the original MovieLens 10M dataset was divided, which are edx and final_holdout_test.

| Model | RMSE |
|---|---|
| Mean_Based Model | 1.05289 |
| Movie_Effect Model | 0.941 |
| Movie+User_Effect Model | 0.85561 |
| Movie+User_Release_Year Model | 0.85529 |
| Movie+User_Release_Year+Genre Model | 0.85498 |
| Regularized Model | 0.85497 |
| Matrix Factorization Model | 0.79894 |
| Final Validation | 0.81853 |

# Chapter 4

# CONCLUSIONS

The development of a movie recommendation system using the MovieLens 10M dataset has followed an incremental approach, testing various models and adjusting parameters to achieve the goal of minimizing the Root Mean Squared Error (RMSE).

Throughout this process, we have gone through different stages that included simple models based on the average, the introduction of specific effects (movies, users, release years, and genres), regularization, and finally matrix factorization. Below, we summarize the most relevant points of the project, highlighting its limitations and proposing possible improvements for future work.

## 4.1 Summary of the Report

- We began with a thorough analysis of the dataset, examining the characteristics of the variables, the distribution of ratings, and potential biases related to users, movies, and genres.
- A basic model based on the average of all ratings was tested, which resulted in a very high RMSE.
- As we added movie and user effects, considerable improvements in the model's accuracy were observed.
- We incorporated release year and genre effects, which further reduced the RMSE, although less significantly.
- Subsequently, regularization was introduced, a key technique to prevent overfitting, improving the model's generalization.
- To further enhance the model, we implemented a matrix factorization model, which allowed us to capture latent interactions between users and movies, achieving notable results in reducing the RMSE.

- The model validation was performed using the final_holdout_ test dataset, confirming that our final model met the goal of achieving an RMSE lower than 0.8649.

## 4.2 Limitations

- Computational Capacity: During the implementation of the matrix factorization model, it was necessary to reduce the dimensions, iterations, and simplify the parameters due to computational limitations. This may have affected the ability to achieve an optimal model.
- Model Complexity: As we added more effects (users, movies, release year, and genres), the complexity of the model increased. Although we achieved improved accuracy, the difference in RMSE was smaller in the later iterations, suggesting that the marginal improvement decreases with each new parameter.
- Limited Regularization: While regularization helped reduce overfitting, the benefits obtained were smaller than expected, indicating that our model was already well-generalized before applying this technique.

## 4.3 Future Work

- Matrix Factorization Optimization: With greater processing capacity, more hyperparameter combinations and iterations could be explored in the matrix factorization model. This could result in a more accurate model, fully leveraging this advanced technique.
- Incorporation of Additional Data: The current model does not include detailed temporal information or potential interactions between genres or users that vary over time. Exploring additional temporal factors, such as the popularity trends of certain genres or movies based on their age, could enrich the model.
- Implementation of Neural Networks: The use of neural networks or deep learning models could be explored, as they are often very effective in recommendation systems. These models have the ability to capture more complex nonlinear interactions between users and movies.

In summary, the project achieved its goal of developing an effective recommendation system with a satisfactory RMSE. However, there is room for improvement in areas such as computational processing optimization and the incorporation of more sophisticated models. These steps would allow for the development of an even more robust and adaptable system for different types of users and movies.

# Chapter 5

# REFERENCES


1. Rafael A. Irizarry. (2019). *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*.
2. Zhu, Hao. (2021). *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*. R package version 1.3.4.
3. González, Javier. (2017). *Building a Recommendation System with R*.
4. Koren, Yehuda, Robert Bell, and Chris Volinsky. (2009). *Matrix Factorization Techniques for Recommender Systems*. *Computer*, 42(8), 30-37.
5. *LaTeX Wikibook*. Retrieved from https://en.wikibooks.org/wiki/LaTeX.