MovieLens

Jean-Marie Roy

2025-12-16

Introduction

Objective

The purpose of this document is to explain how to create a movie recommendation system using the MovieLens dataset. During the PH125.8x: Data Science: Machine Learning course, the dataset used was from the the dslabs package. For this exercise, we will use a 10M rows dataset available here :

> https://grouplens.org/datasets/movielens/10m/

The history and context of this dataset are available here :

> F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

First we will setup the environment and download and generate the dataset, then we will analyse the data, and finally we will train a machine learning algorithm using the inputs in the first subset (edx) to predict the movie ratings in the validation set (final_holdout_test).

The goal of this exercise is to be able to predict the movie ratings in the test set with a root mean square error (RMSE) lower than **0.86490**.

This task is inspired by the Netflix challenge, which aimed to predict ratings without utilizing any user data (such as age or gender) due to privacy concerns. Ultimately, the goal is to predict which films a user would enjoy based on their previous ratings.

Data preparation

This section explains how the data is imported and prepared for further analysis. A portion of the process is based on the code provided in the course, specifically within the “01 Setup.R” script.

The code processes the MovieLens 10M dataset to create training (edx) and testing (final_holdout_test) datasets suitable for analysis.

It begins by checking for necessary libraries, downloading the dataset, and extracting the ratings and movies files if they aren’t already present. It reads and cleans the data, ensuring that columns are correctly formatted.

The main dataset is created by merging ratings with movie details, and then a random subset is taken as a test set—10% of the data. To maintain consistency, only those users and movies present in the training set are included in the final test set. Any excluded rows are then added back to the training set. Finally, the code cleans up by removing unnecessary variables.

Exploratory Data Analysis

Structure

This section provides an overview of the datasets used in our analysis. We have two datasets: one for training our prediction algorithm and another for testing its effectiveness.

The training dataset, edx, shares the same structure as the final_holdout_test dataset, which is used for testing.

The edx dataset contains approximately 9 million ratings for various movies and consists of six variables :

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09	NA	NA
3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

The table below outlines the variables present in the dataset :

Variable	Description
`userId`	User ID (anonymised)
`movieId`	Unique movie ID
`rating`	Rating from 0 to 5 including half numbers (1.5, 2.5. . .)
`timestamp`	Date and time the rating was created. It is stored as a number of seconds since the 1st of January 1970
`title`	Title of the movie with the year it was released, in the form “name (year)”
`genres`	Genre of the movie (action, drama. . .)

From the dataset, we can determine the total counts of movies, users, and ratings :

Dataset	Type	Movies	Users	Ratings
edx	Train	10,677	69,878	9,000,055
final_holdout_test	Test	9,809	68,534	999,999

The analysis reveals that approximately 10,000 movies were rated by 70,000 users, resulting in a total of 9,000,000 ratings.

The pair (movieId, userId) can be used as a primary key, as each user has rated a specific movie only once :

Dataset	(movieId, userId)	Ratings
edx	9,000,055	9,000,055
final_holdout_test	999,999	999,999
Overlap	0	

We can also observe that a rating given by a user for a specific movie cannot appear in both the training and test datasets.

Data Transformation

The first few entries of the edx dataset suggest potential transformations for enhanced analysis :

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy;Romance
2	1	185	5	838983525	Net, The (1995)	Action;Crime;Thriller
4	1	292	5	838983421	Outbreak (1995)	Action;Drama;Sci-Fi;Thriller
5	1	316	5	838983392	Stargate (1994)	Action;Adventure;Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action;Adventure;Drama;Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children;Comedy;Fantasy

The timestamp variable can be decomposed into various components, including the day of the week, day, month, year, and hour (minutes and seconds are likely unnecessary for our analysis).

Additionally, the title variable contains the release year formatted as "title (year)." We can extract both the title without the year and the year of release.

While we can also separate the genres, we are not permitted to add rows to the dataset, only columns, so this transformation is not feasible.

The table below displays the modified edx dataset, which retains the genres column but omits it from this view due to space constraints :

userId	movieId	rating	t_day_of_week	t_day	t_month	t_year	t_hour	title	year
1	122	5	5	2	8	1996	11	Boomerang	1992
1	185	5	5	2	8	1996	10	Net, The	1995
1	292	5	5	2	8	1996	10	Outbreak	1995
1	316	5	5	2	8	1996	10	Stargate	1994
1	329	5	5	2	8	1996	10	Star Trek: Generations	1994
1	355	5	5	2	8	1996	11	Flintstones, The	1994

This allows us to focus on the other relevant attributes for analysis while keeping the genres information in the dataset.

Data Analysis

Movies

The top five movies with the highest number of ratings each received 30,000 ratings, indicating that half of the users rated these films :

index	title	year	count
1	Pulp Fiction	1994	31,362
2	Forrest Gump	1994	31,079
3	Silence of the Lambs, The	1991	30,382
4	Jurassic Park	1993	29,360
5	Shawshank Redemption, The	1994	28,015

Movie Ratings Visualization

The left figure illustrates that a small number of movies received the majority of ratings, while most received very few.

To better highlight this trend, we can apply a log10 transformation to the y-axis. This transformation reveals that although most movies garnered at least 10 ratings, only half exceeded 100 ratings, as demonstrated in the right figure :

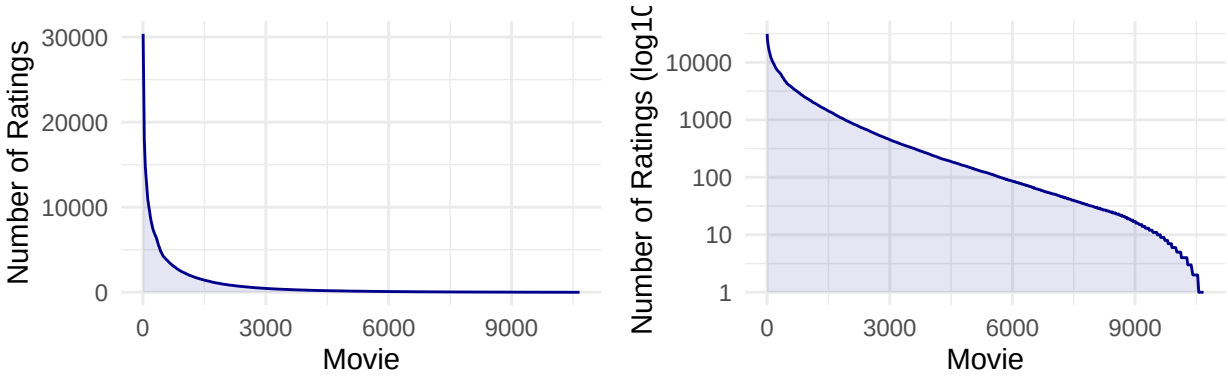

Figure 1: Number of ratings / Movie

Movie Release and Ratings Trends

The figure on the left shows that the number of movies released per year increased significantly during the 1980s.

Similarly, the figure on the right demonstrates that the number of ratings received per year for newly released movies also rose during this period. However, it appears that older movies received significantly fewer ratings compared to more recent releases :

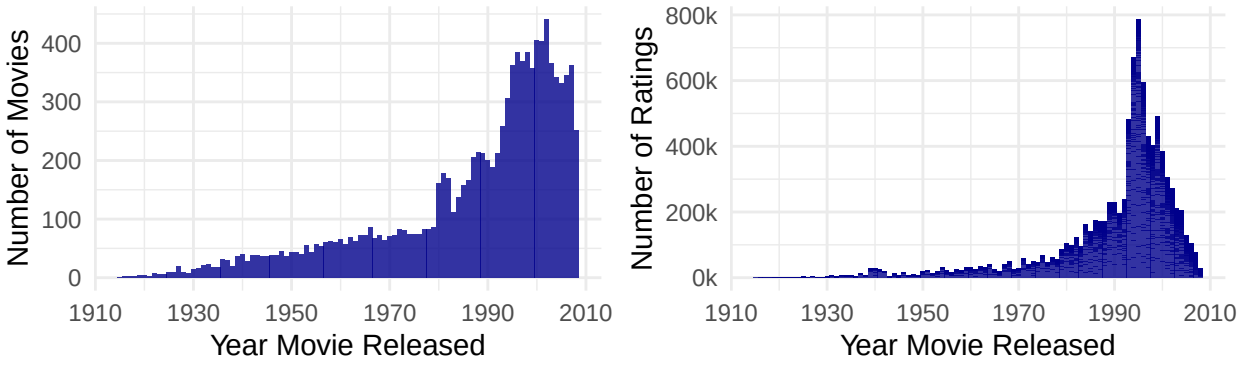

Figure 2: Number of Movies Released per Year and Number of Ratings per Year

Users

As previously mentioned, the edx dataset contains approximately 70,000 users. Below are the top five users with the highest number of ratings :

index	count
1	6,616
2	6,360
3	4,648
4	4,036
5	4,023

A small number of users account for a large proportion of ratings, as shown in the figure on the left, which illustrates that the majority of ratings come from a minority of users.

By applying a log10 transformation to the y-axis, we can see that all users rated at least 10 movies, while only one-third rated more than 100 movies, as depicted in the figure on the right :

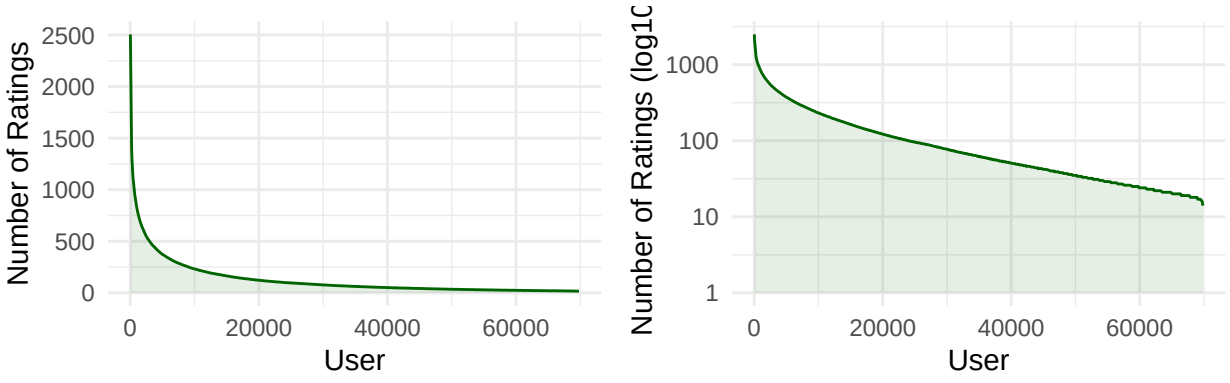

Figure 3: Number of ratings / User

Ratings

The figures below demonstrate that the distribution of ratings can be approximated as normal when rounded :

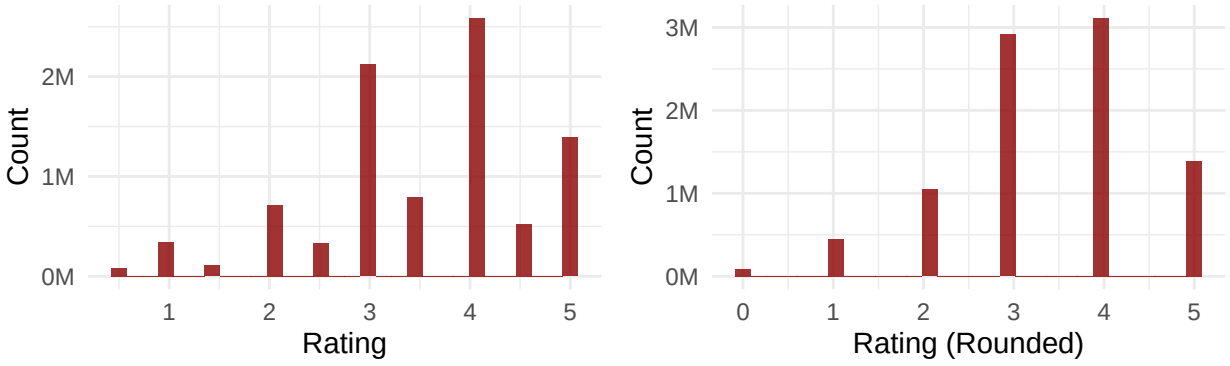

Figure 4: Number of ratings

The table below displays the average and standard deviation :

Type	Value
Average	3.512465
Standard Deviation	1.060331

Variables

List of variables and their distinct value counts :

Variable	Distinct_Count
userId	69,878
movieId	10,677
title	10,407
genres	797
year	94
t_day	31
t_hour	24
t_year	15
t_month	12
t_day_of_week	7

Distribution of Average Ratings per Variable

We will consider only those variables that have a sufficient number of distinct values : userId, movieId, title and genres.

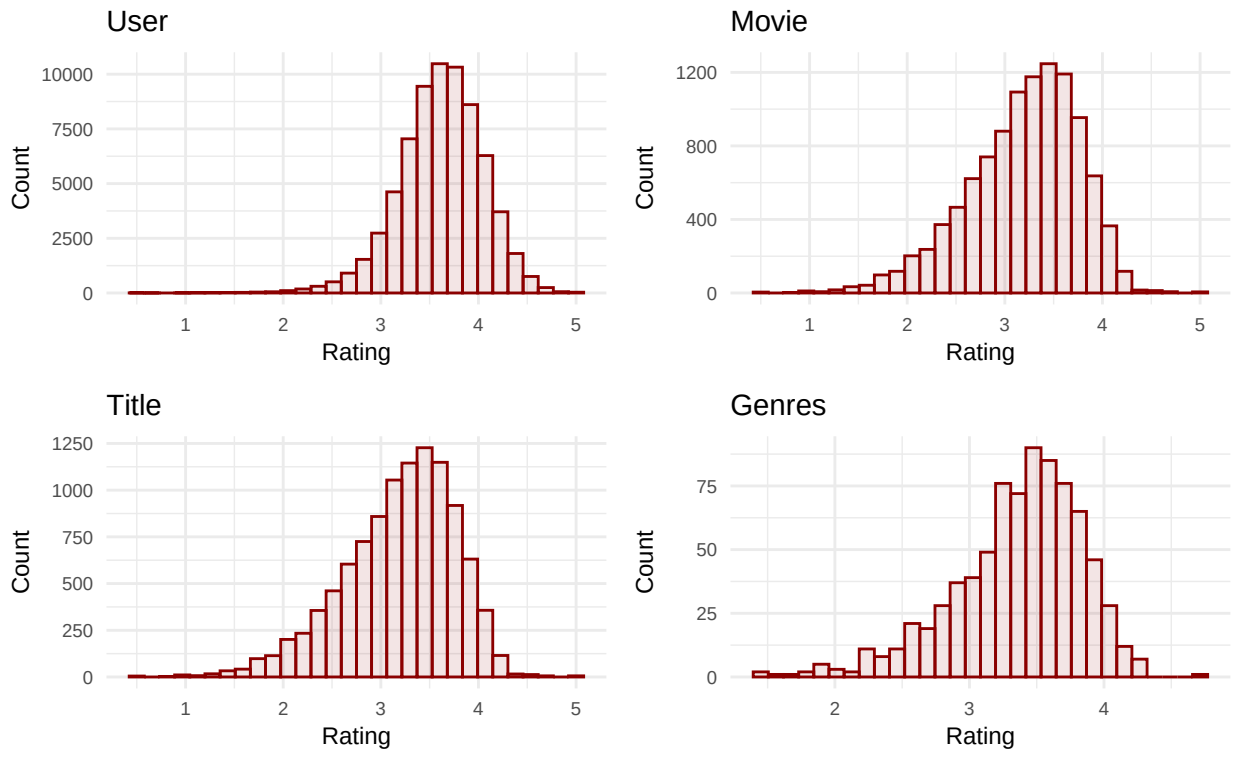

Figure 5: Distribution of ratings

We will now compare Title against Movie as the graphs look very similar :

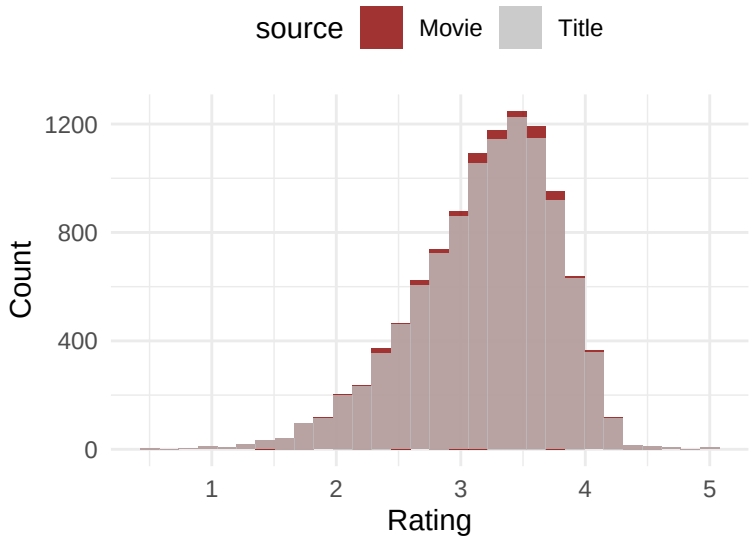

Figure 6: Comparison of the Distribution of ratings

For the upcoming analysis, we will focus exclusively on the movie, user, and genres variables, which have a high number of unique values, while excluding the title variable.

Variables with Low Distinct Values

We will now examine variables that have a low number of distinct values, specifically the timestamp information and the year of release.

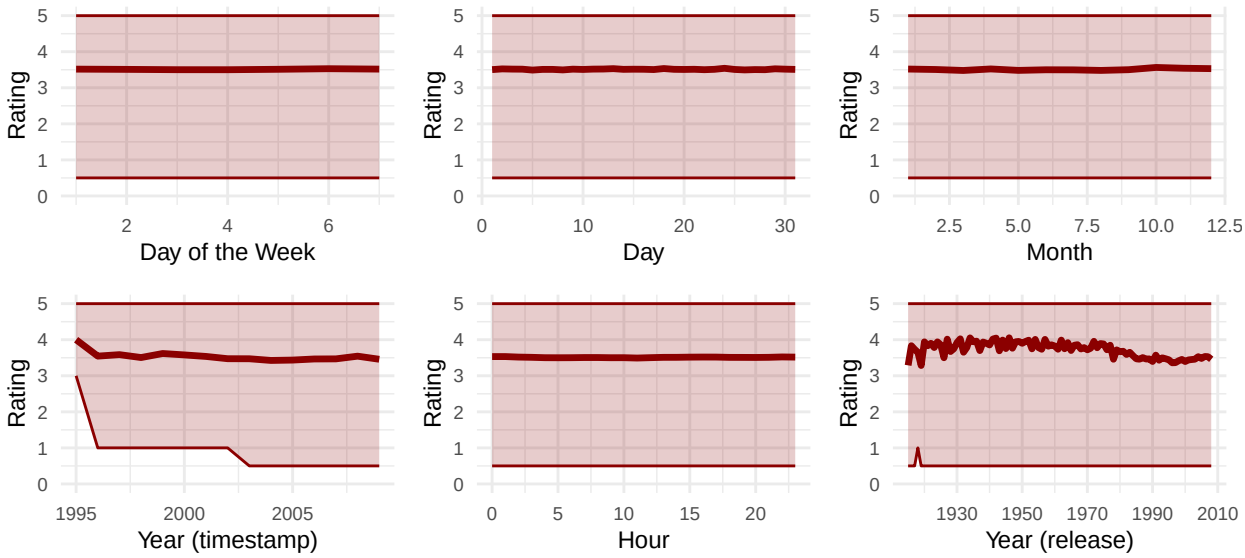

Figure 7: Distribution of ratings per variable

As observed, only the timestamp (year) and year of release exhibit sufficient variation (considering both the mean and minimum values for the timestamp) to be included in the forthcoming analysis.

Predictions

Given the big number of ratings (9 millions) we won't use any training algorithm such as lm as they would require an extensive time to compute. We will try to estimate the ratings using average and biases.

The goal of this exercise is to get the minimum root mean square error (RMSE). The RMSE can be defined as :

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}\left(\hat{y}_i - y_i\right)^2} \tag{1}$$

where :
y_i : The observed values (actual ratings).
$\hat{y}_i$: The predicted values (model outputs).
N: The number of observations.

Target

The target RMSE we want to achieve (get a lower value) is : RMSE = 0.86490

Average

The average for the predictions is : $\mu = 3.512465$ If we choose a constant number for the predictions, then μ will be the one which will give the minimum RMSE :

$$\hat{y}_i = \mu \tag{2}$$

The RMSE will then be : RMSE = 1.060331

The error can be defined as :

$$\epsilon_i = \hat{y}_i - y_i \tag{3}$$

The following plots illustrate the distribution of ratings in descending order, with the average value represented by a horizontal orange line. The plot on the right depicts the distribution of errors :

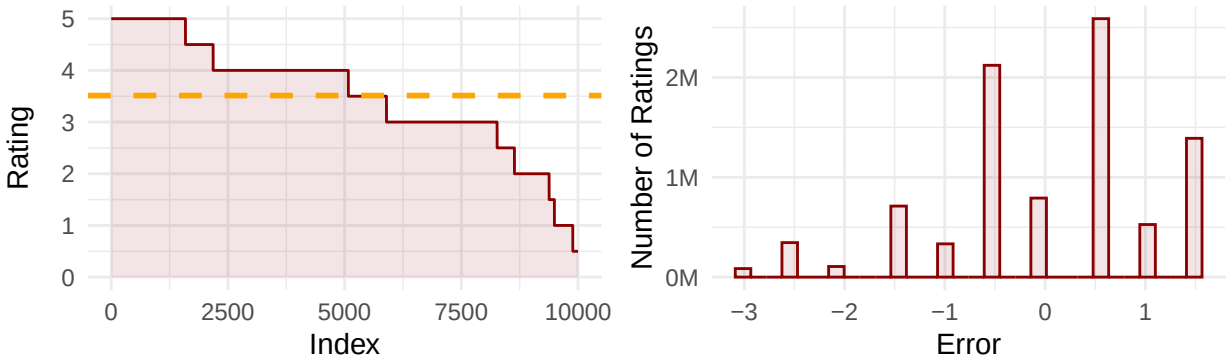

Figure 8: Prediction (average)

Bias (simple)

The next step is to introduce a bias so the prediction can be written :

$$\hat{y}_i = \mu + b_{v,i} \tag{4}$$

with :

$$b_{v,i} = \frac{1}{N} \sum_j (y_j - \mu) \tag{5}$$

where :
$\hat{y}_i$: The predicted values for a variable v (model outputs).
$b_{v,i}$: The bias for a variable v.

The figure below illustrates our objective. For each movie (or user or genre), we calculate the average rating. In this example, there are three movies, each represented by different colors, with individual points indicating their ratings. The line or segment displays the average rating for each specific movie (or user or genre). The orange line shows the global average :

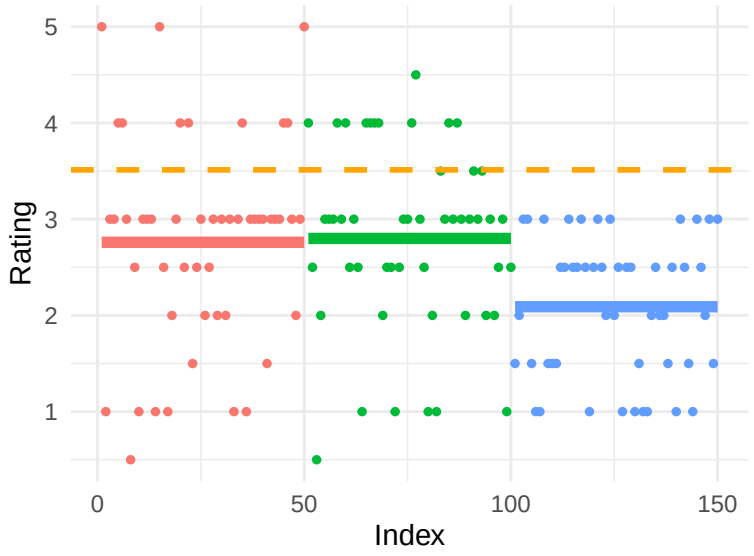

Figure 9: Bias

During the prediction process, we will default to the overall average rating. We will then adjust this by adding the difference between the average rating for each movie (or user or genre) and the global average μ, with this difference representing the bias.

If the same movie (or user or genre) appears in the test dataset, we will use its average rating. Otherwise, we will predict the global average from the training dataset. This approach should result in a lower RMSE.

The table below presents the RMSE values for different prediction types used in our analysis : average and bias for Movie, User, Genres, Year(timestamp) and Year (release).

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
User	0.9700086
Genres	1.0179838
Year (timestamp)	1.0585990
Year (release)	1.0493440

- **Target**: The RMSE for the target model is 0.8649000, indicating the maximum value of the RMSE we want to get.
- **Average**: The baseline RMSE calculated using the global average is 1.0603313, which serves as a reference point.
- **Movie**: When predictions are made based on individual movie averages, the RMSE is 0.9423475, showing improved accuracy compared to the average.
- **User**: Prediction based on user-specific averages yields an RMSE of 0.9700086, which is slightly higher than the movie-based predictions.

- **Genres**: Using genre averages results in an RMSE of 1.0179838, indicating diminished accuracy compared to both movie and user averages.
- **Year (timestamp)**: The RMSE for the year based on timestamp data stands at 1.0585990, showing median accuracy.
- **Year (release)**: Similarly, predictions based on the release year have an RMSE of 1.0493440, which also indicates average performance.

In summary, the table demonstrates that the predictions based on movies yield the best results among the other features considered.

The following four plots illustrate the distribution of errors for the biases associated with movies, users, genres, and the year (timestamp). We have excluded the release year, as its RMSE is very similar to that of the timestamp year. Ideally, we want the errors to be as close to zero as possible. The plots clearly indicate that the timestamp year yields the highest errors, while the genre bias performs better but still falls short. The movie bias demonstrates the best performance, followed closely by the user bias.

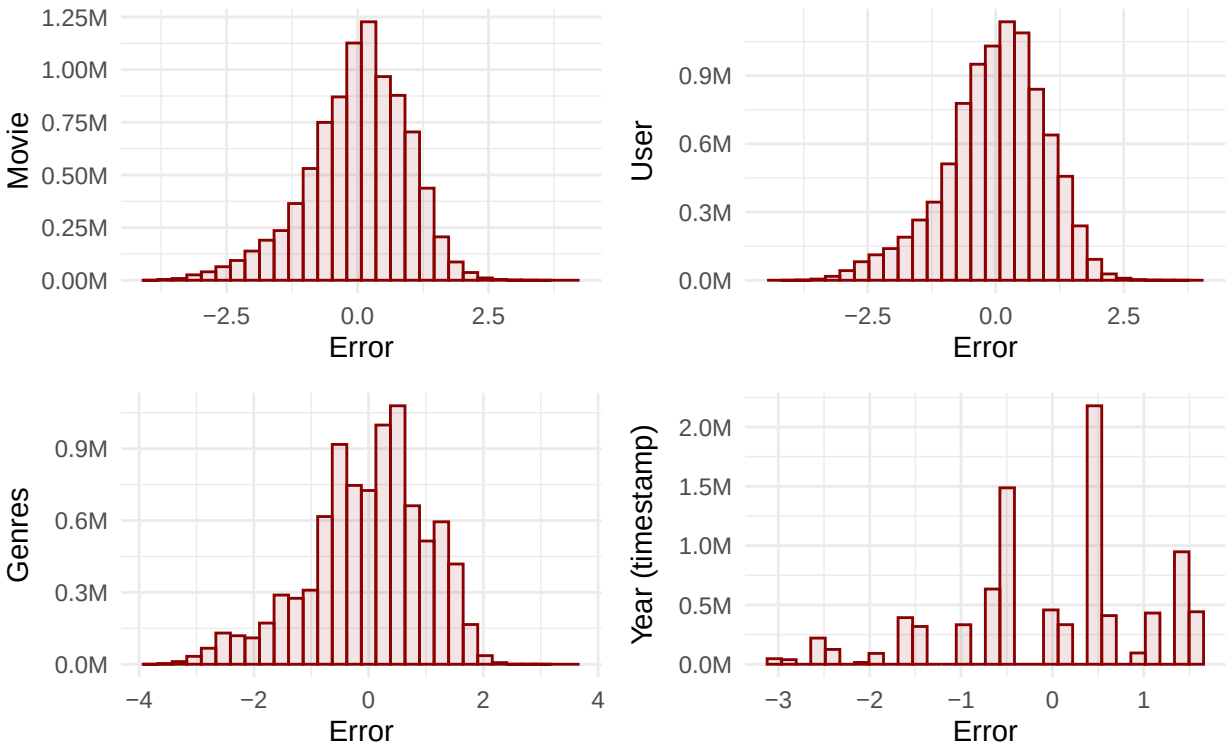

Figure 10: Distribution of error (Bias)

As an example, some predictions for the movie bias are shown in the table below :

userId	movieId	title	year	rating	pred_1
64957	1682	Truman Show, The	1998	3.5	3.767434
27168	780	Independence Day (a.k.a. ID4)	1996	4.0	3.376903
40876	593	Silence of the Lambs, The	1991	5.0	4.204101
58604	2478	Three Amigos	1986	4.0	3.045719
8622	413	Airheads	1994	3.0	2.767378

Bias (Multiple)

The next step is to introduce a second bias so the prediction can be written :

$$\hat{y}_{v,w,i} = \mu + b_{v,i} + b_{w,i} \tag{6}$$

where :
$\hat{y}_{v,i}$: The predicted values for a variable v (model outputs).
$b_{v,i}$: The bias for a variable v.
$b_{w,i}$: The bias for a variable w.

We can keep the same formula (5) for the 2 biases, which will give a better RMSE for the movie and user variables : **0.8767534**

However, we can even fine tune the calculation of the bias. If we keep the same formula as (5) for the first bias, we can calculate the next one with this formulas :

$$b_{w,i} = \frac{1}{N} \sum_{j} (y_j - \mu - b_{v,j}) \tag{7}$$

The first bias is defined as the difference between the overall average and the average for a specific movie (or user or genre).

However, the definition of the second bias has changed. When the first bias pertains to the movie, the second bias—applicable to the user—will be the difference between the average rating for a specific user and the average of the predicted ratings for the movies that this user has rated.

The table below presents the RMSE values for different prediction types used in our analysis : average and bias for Movie, and the multiple biases applied to Movie and User, and all variables : Movie, User, Genres, Year(timestamp) and Year (release).

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Movie + User	0.8567039
All	0.8561090

The following plot show the distribution of errors for the Movie bias (simple) and the Movie + User one (multiple) :

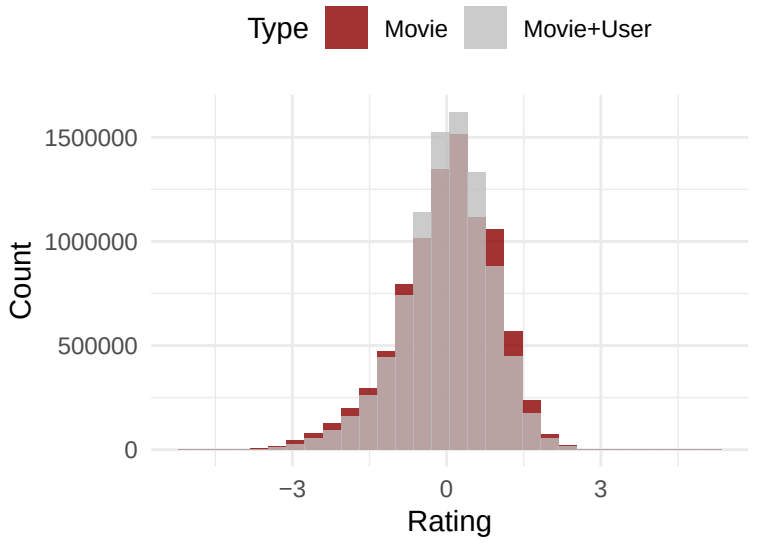

Figure 11: Comparison of the Distribution of errors

As an example, some predictions for the movie and user biases are shown in the table below :

userId	movieId	title	year	rating	pred_1	pred_2
64957	1682	Truman Show, The	1998	3.5	3.767434	3.488565
27168	780	Independence Day (a.k.a. ID4)	1996	4.0	3.376903	3.110990
40876	593	Silence of the Lambs, The	1991	5.0	4.204101	4.321057
58604	2478	Three Amigos	1986	4.0	3.045719	2.807479
8622	413	Airheads	1994	3.0	2.767378	2.781811

Regularisation

Regularization adjusts the model's bias by incorporating penalty terms into the loss function. This adjustment helps to tune the model's complexity, making it more robust against overfitting. By adding penalties to large coefficients, regularization reduces the influence of outliers. This results in a model that doesn't react excessively to extreme values in the data.

The bias formulas for the movies and the users can now be written :

$$b_m = \frac{\sum_j (y_j - \mu)}{N + \lambda} \tag{8}$$

and :

$$b_{u,m} = \frac{\sum_j (y_j - \mu - b_m)}{N + \lambda} \tag{9}$$

The figure below shows how the (RMSE varies with the regularisation parameter λ :

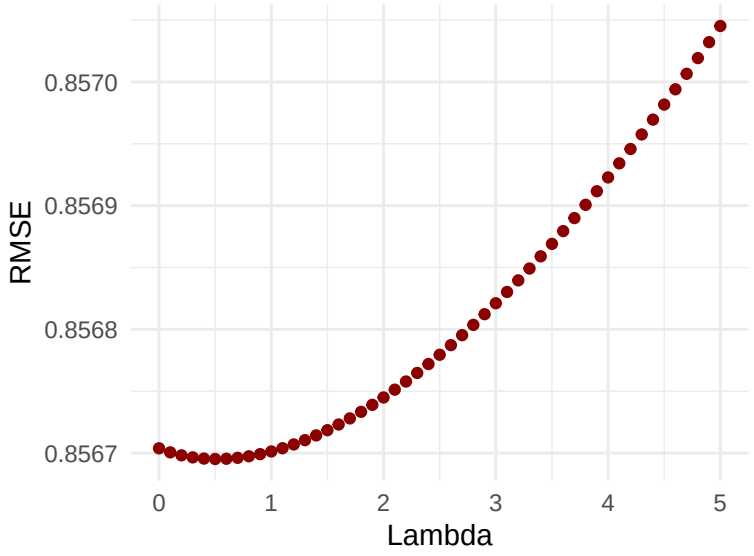

Figure 12: Cross-validated RMSE vs Regularisation Strength

The minimum value is reached when $\lambda = 0.5$, the RMSE is then **0.8566952**.

This result is higher than with the previous method and not really below the Movie + User method :

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Movie + User	0.8567039
All	0.8561090
Regularisation (Movie + User)	0.8566952

Multi-dimension bias

The idea is to use a couple of variables to calculate the bias.

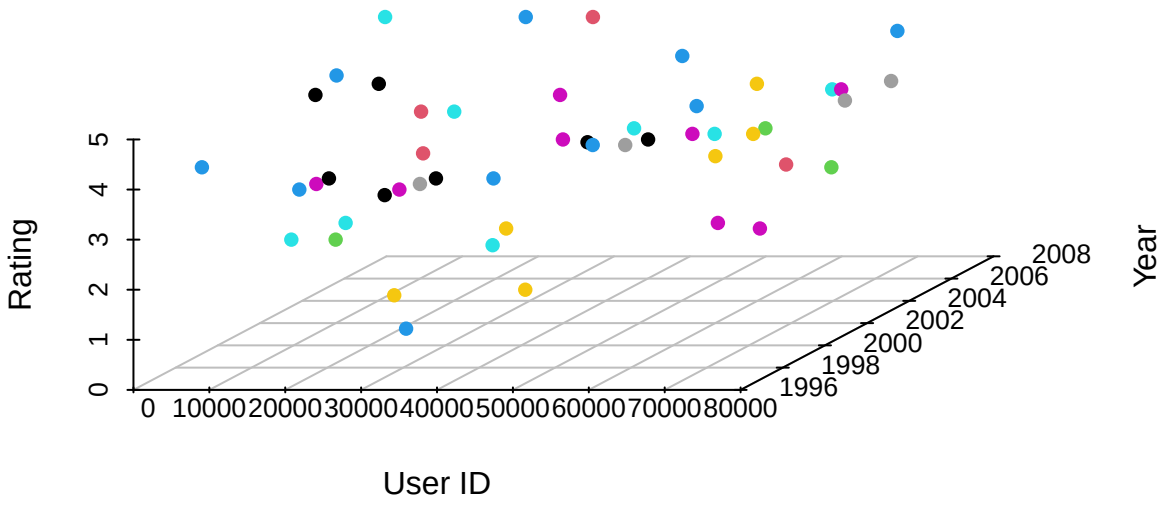

Figure 13: Rating per user, year (timestamp) and movie

The table below displays for each couple of variables the number of rows in final_holdout_test not in the edx dataset :

Combination	Nb Rows
userId, movieId	999,999
userId, title	994,147
userId, genres	336,086
userId, year	73,053
movieId, t_year	704
t_year, title	692
userId, t_year	162
t_year, genres	16
t_year, year	1
movieId, title	0
movieId, year	0
movieId, genres	0
title, year	0
title, genres	0
year, genres	0

The more rows only in final_holdout_test, the better RMSE we will get in the training set, but we will

probably also get a high RMSE in the test set.

We can even treat three variables as a triplet. The table below shows, for each combination of these three variables, how many rows in final_holdout_test are absent from the edx dataset :

Combination	Nb Rows
userId, movieId, t_year	999,999
userId, movieId, title	999,999
userId, movieId, year	999,999
userId, movieId, genres	999,999
userId, title, year	999,996
userId, title, genres	998,705
userId, t_year, title	995,577
userId, year, genres	761,808
userId, t_year, genres	377,194
userId, t_year, year	93,965
movieId, t_year, title	704
movieId, t_year, year	704
movieId, t_year, genres	704
t_year, title, year	704
t_year, title, genres	704
t_year, year, genres	188
movieId, title, year	0
movieId, title, genres	0
movieId, year, genres	0
title, year, genres	0

As we saw earlier, we had the lowest RMSE by applying a bias on the movie variable before the user one, so we will continue with the couples (movieId, t_year), (userId, t_year) and then (t_year, year, title).

To further reduce the RMSE, we will also apply a regularization with the same value of λ as found earlier (0.5).

The formulas for the 3 bias will then be :

$$b_v = \frac{\sum_j (y_j - \mu)}{N + \lambda} \tag{10}$$

$$b_w = \frac{\sum_j (y_j - \mu - b_v)}{N + \lambda} \tag{11}$$

$$b_x = \frac{\sum_j (y_j - \mu - b_v - b_w)}{N + \lambda} \tag{12}$$

The following R code performs bias calculations based on the edx_movies dataset and joins the results back into the dataset:

```r
# Calculate bias for each movie and year, and join to the edx_movies dataset
b_v <- edx_movies %>%
  group_by(movieId, t_year) %>%
  summarise(b_v = sum(rating - mu)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_v, by = c("movieId", "t_year")) %>%
  mutate(b_v = replace_na(b_v, 0))

# Calculate bias for each user and year, and join to the edx_movies dataset
b_w <- edx_movies %>%
  group_by(userId, t_year) %>%
  summarise(b_w = sum(rating - mu - b_v)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_w, by = c("userId", "t_year")) %>%
  mutate(b_w = replace_na(b_w, 0))

# Calculate bias for each title per year, and join to the edx_movies dataset
b_x <- edx_movies %>%
  group_by(t_year, year, title) %>%
  summarise(b_x = sum(rating - mu - b_v - b_w)/(n() + lambda))

edx_movies <- edx_movies %>%
  left_join(b_x, by = c("t_year", "year", "title")) %>%
  mutate(b_x = replace_na(b_x, 0))
```

After incorporating the three bias components, the model achieves an RMSE of **0.8417134** on the train set.

Type	RMSE
Target	0.8649000
Average	1.0603313
Movie	0.9423475
Movie + User	0.8567039
All	0.8561090
Regularisation (Movie + User)	0.8566952
2D/3D Interaction	0.8417134

As an example, some predictions are shown in the table below :

userId	movieId	title	year	rating	pred_1	pred_2	pred_3
64957	1682	Truman Show, The	1998	3.5	3.767434	3.488565	3.369805
27168	780	Independence Day (a.k.a. ID4)	1996	4.0	3.376903	3.110990	2.765118
40876	593	Silence of the Lambs, The	1991	5.0	4.204101	4.321057	4.348854
58604	2478	Three Amigos	1986	4.0	3.045719	2.807479	2.990552
8622	413	Airheads	1994	3.0	2.767378	2.781811	2.716821

Clamping

While we have now a relatively low RMSE of 0.8417290, further analysis of the data shows that the predicted values range from -1 to 6.2 when the ratings range from 0.5 to 5 :

Variable	Min	Max
Rating	0.500000	5.000000
Prediction	-1.036935	6.204538

We should be able to reduce the RMSE by replacing all values out of the expected range by 0.5 or 5. We can even go further by replacing all values outside a smaller range by 0.5 or 5.

Let's try first with the maximum value a prediction can take. The graph below shows that the minimum RMSE we can get is when we clamp all data above 4.64 :

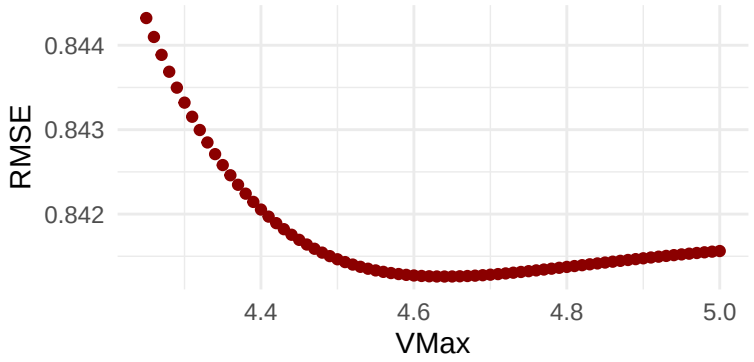

Figure 14: RMSE vs VMax

Using a maximum value of 4.64, we can continue and determine the minimum value which will give us the best RMSE is 0.93 :

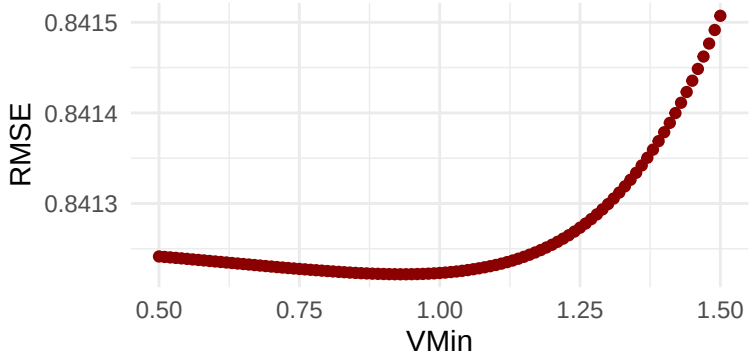

Figure 15: RMSE vs VMin

We now have an RMSE of **0.8412219**.

Final Test

We now apply the latest prediction algorithm to the final_holdout_test data set.

Join the bias terms

```
final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_v, by = c("movieId", "t_year")) %>%
  mutate(b_v = replace_na(b_v, 0))

final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_w, by = c("userId", "t_year")) %>%
  mutate(b_w = replace_na(b_w, 0))

final_holdout_test_movies <- final_holdout_test_movies %>%
  left_join(b_x, by = c("t_year", "year", "title")) %>%
  mutate(b_x = replace_na(b_x, 0))
```

Compute the predictions

The predicted rating is the global mean (μ) plus the three bias components. The result is then clamped to the admissible rating range (0.93 – 4.64).

```
final_holdout_test_movies$pred <- pmin(pmax(mu +
                                            final_holdout_test_movies$b_v +
                                            final_holdout_test_movies$b_w +
                                            final_holdout_test_movies$b_x, 0.93), 4.64)
```

Evaluate the model

The Root Mean Squared Error (RMSE) is computed by comparing the observed ratings with the predicted values.

```
RMSE(final_holdout_test_movies$rating, final_holdout_test_movies$pred)
```

The final RMSE obtained on the hold-out test set is **0.8579873**.

Conclusion

We achieved our target, obtaining an RMSE of **0.857987**, which is lower than the required maximum of **0.86490**.

To reach this result we experimented with several bias-based methods and selected the combination that performed best on the validation data.

Further improvements are possible. The distribution of ratings in the training set is not perfectly normal because a subset of users consistently give half-star ratings. If we separate the data into half-mark and full-mark subsets, each subset approximates a normal distribution.

A promising next step would be a two-stage modeling approach:

Stage 1 – Classification: Predict whether a user will assign a half-mark or a full mark to a given movie. This binary decision could be modeled with logistic regression, a tree-based classifier, or any suitable machine-learning algorithm.

Stage 2 – Regression: Apply separate bias-adjusted regression formulas for the two groups (one for half-marks, another for full marks). Each regression would use the appropriate bias terms (movie, user, time, etc.) tailored to that subgroup.

By first determining the likely rating granularity and then applying a specialized regression model, we expect to reduce prediction error further.