MovieLeans Project

HarvardX: PH125.9x Data Science: Capstone

Rebello, Carlos E. Piffer

November, 2020

Contents

1 Executive Summary

This project is about a movie recommendation system developed based on the ratings previously given by users. This type of system is very common in machine learning algorithms and is widely used in companies with a large database of products and customers who are encouraged to register their ratings.

The goal here is to train a machine learning algorithm that provides user ratings, between 0.5 and 5 stars.

The value used to evaluate the performance of the algorithm is Root Mean Square Error (RMSE).

RMSE

It is a measure of accuracy for comparing forecasting errors from different models for a given dataset. In this case, it is the typical mistake we make when predicting a movie rating.

If it is greater than 1, our typical error is greater than a star, which is not a good thing. We define $y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$. The RMSE is then defined as:

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

The RMSE resulting from this algorithm must be less than 0.8649.

For this project was used the database made available by the team through the link

http://files.grouplens.org/datasets/movielens/ml-10m.zip,

which was divided into edx (training set) and validation (test set used only for final model validation). The dataset could also be downloaded separately:

https://www.dropbox.com/s/nspymeso8rmmak1/edx.rds?dl=1
https://www.dropbox.com/s/x0s477b0kzxpl6i/validation.rds?dl=1

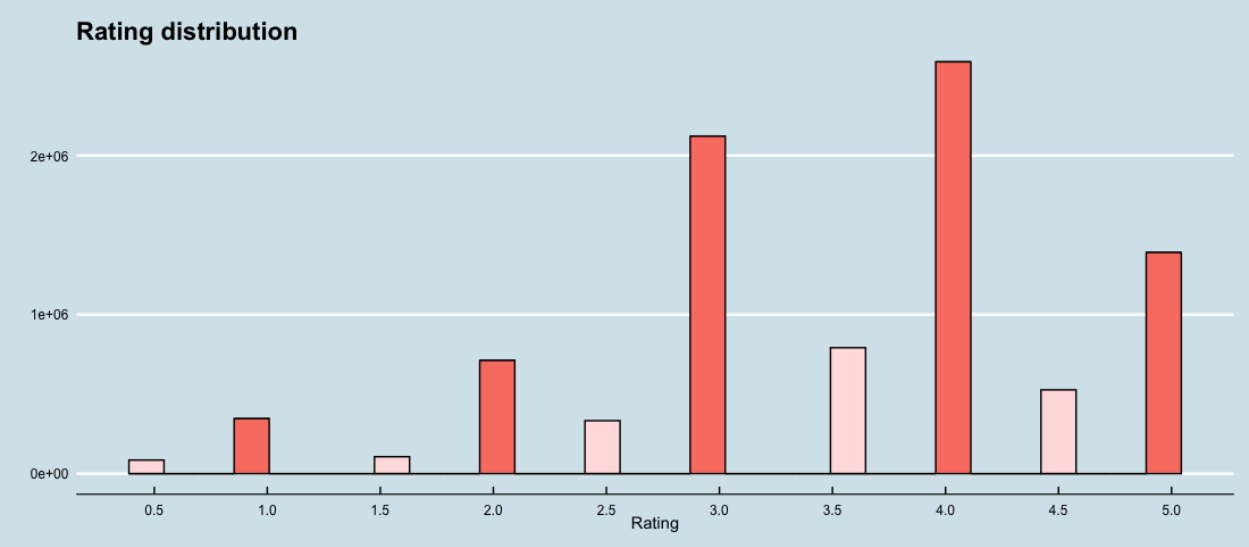
It contains six variables: userId, movieId, rating, timestamp, title and genres. Each line represents a single user rating for a single movie with a total of 9,000,055 observations, 69,878 unique users and 10,677 different movies. The validation set represents 10% of the 10m MovieLens dataset with a total of 999,999 occurrences.

To develop the algorithm and define the parameters that minimize the RMSE, cross validation was used, so edx was divided into train_set, with 7,200,043, and test_set, with 1,799,974 observations (20% of edx).

The model considered an overall average of ratings and the bias of each user, movie and genre. The year movies were released and rated were also considered. In addition, a penalty was given to avoid outliers effects of movies with few rates and regressions were used to predict residuals with a smaller data set. After all, the average of chosen techniques was used for the final prediction.

2 Methods and Analysis

Observing the general distribution of ratings, we notice the grades 4, 3 and 5 are the most common. We also notice that evaluations with a half star are unusual when compared to whole stars, only 20.48%.

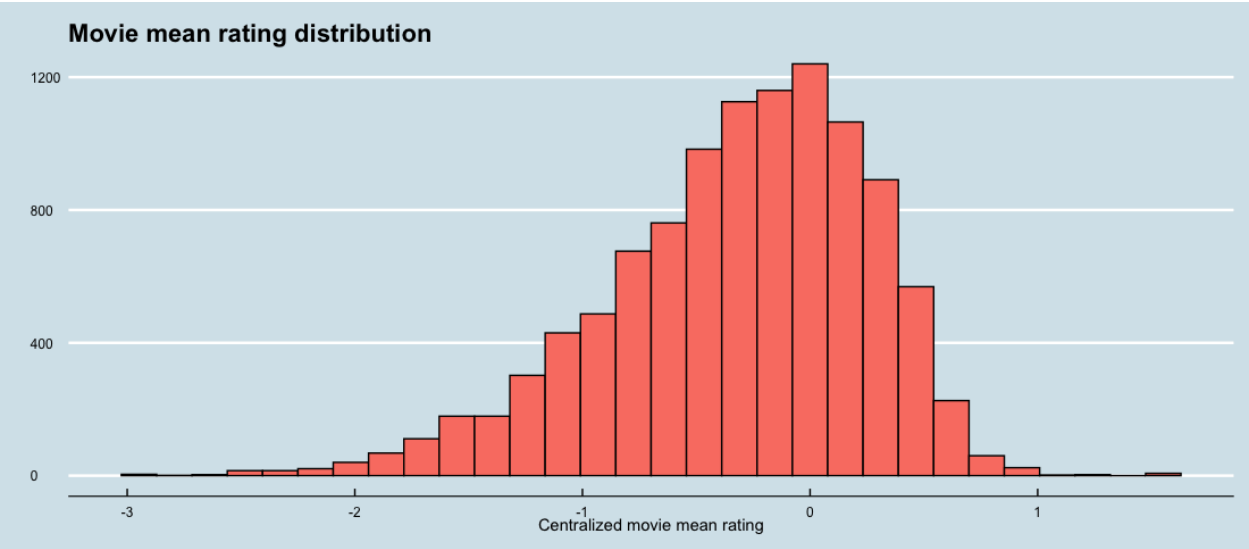


The overall ratings is 3.51 and, if we use it as a single prediction we will have an RMSE = 1.0604863

The first step was to centralize the rating column by subtracting the average (μ), then finding the variables that could explain the residual.

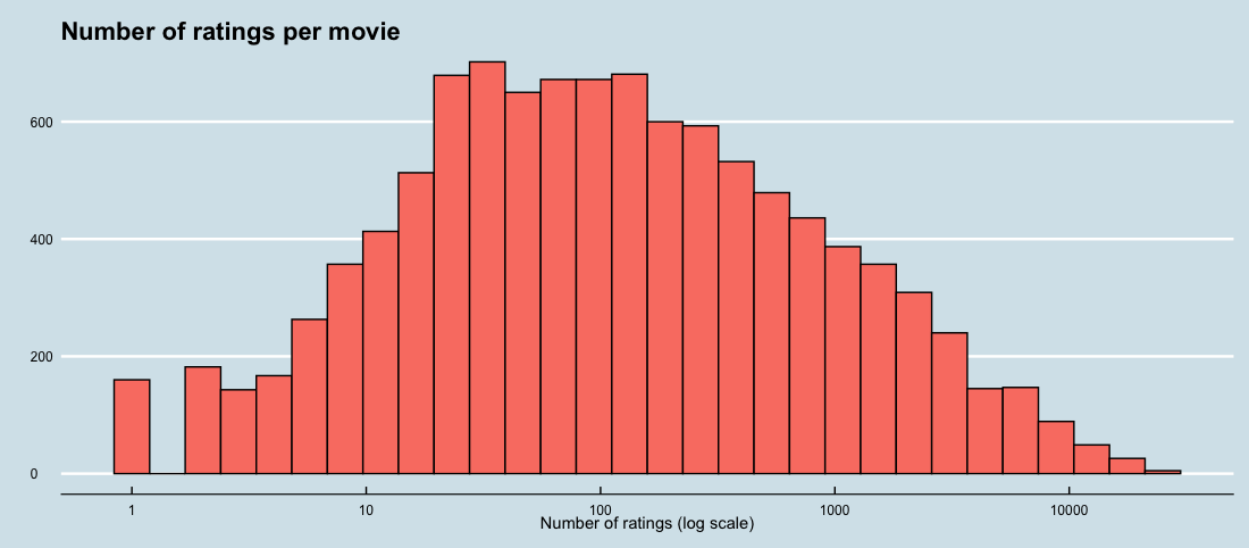
2.1 Movie effect

Looking at the data we notice some movies are better rated than others and residual follows a distribution close to normal. This seems natural, since some productions and scripts are better than others.



To improve the model, we add each movie bias (b_i), which is its average residual.

At the train set movies have an average of 676 reviews. While some movies have more than 20,000 reviews, 160 have only one.



It is very risky to say a title has an average rating of 5 stars based on only one user opinion, so we had to penalize movies with few ratings and this was made by regularization.

Regularization

The process basically consists of adding a constant (λ) to each movie number of observations (n_i) when calculating the average. If n_i is large λ has no effect, since $n_i + \lambda \approx n_i$. On the other hand, if n_i is small b_i tends to 0 and will be calculated with the following equation:

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

And our forecast is taken with:

$$\hat{y} = \hat{\mu} + \hat{b}_i$$

and

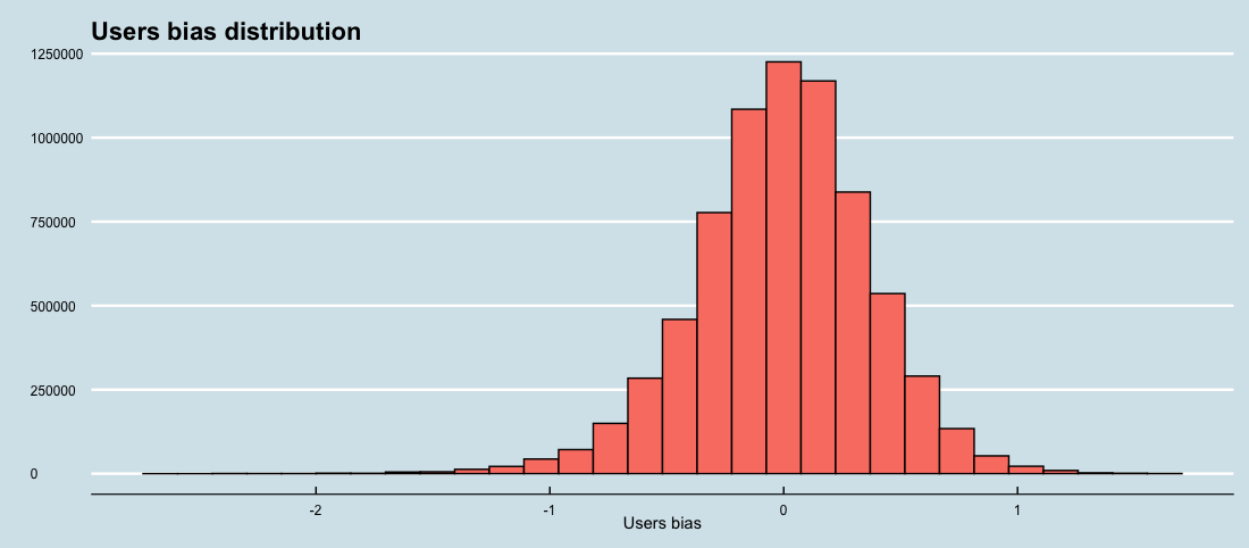
$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i}$$

Where $\varepsilon_{u,i}$ is the error for user u and movie i with a zero average. The values less than 0.5 and greater than 5 were brought into the margin, improving the results.

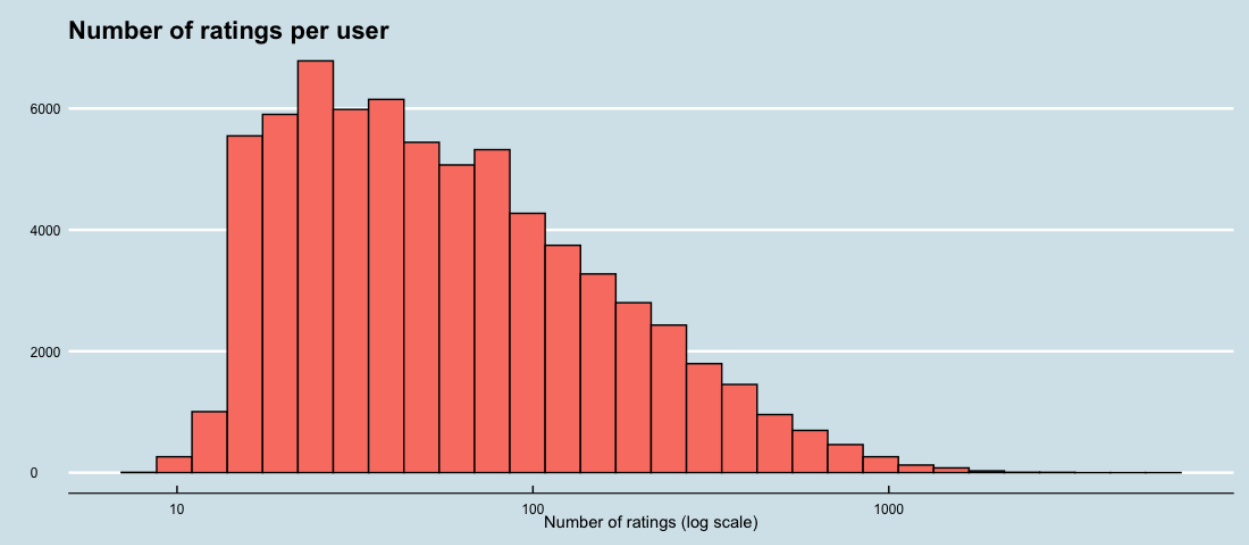
The RMSE obtained now is 0.9438568, better than the previous one.

2.2 User effect

Users also have their own bias. Some tend to give good grades always, while others tend to give below average rates.



Was also added the user effect, represented by b_u and, as some users rated very few movies, which makes estimating the bias unreliable,



we have to use regularization.

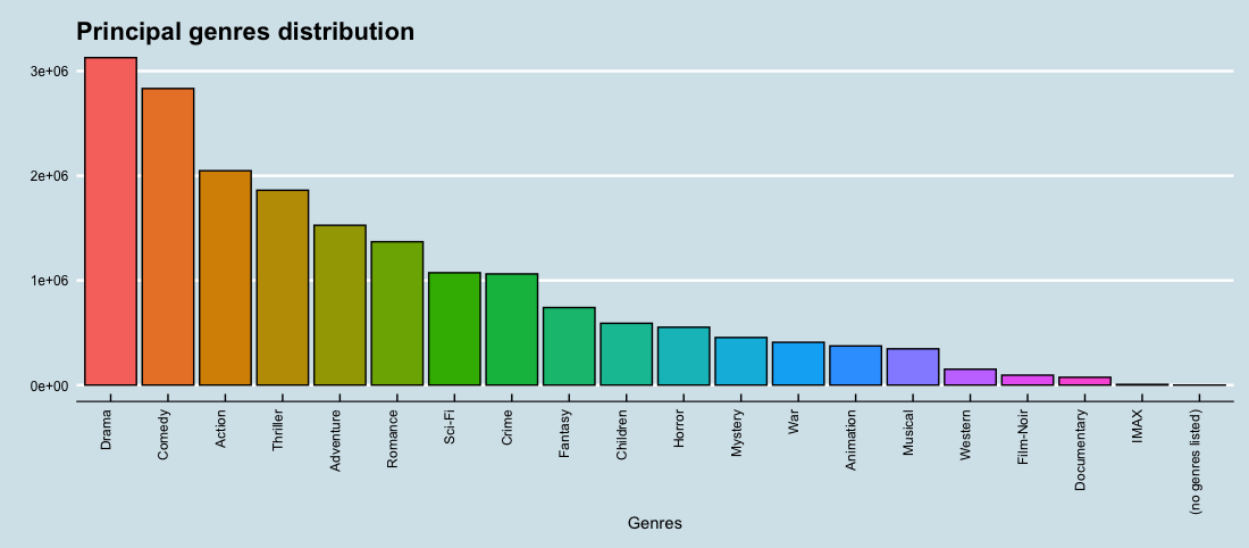
$$b_u = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i)$$

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

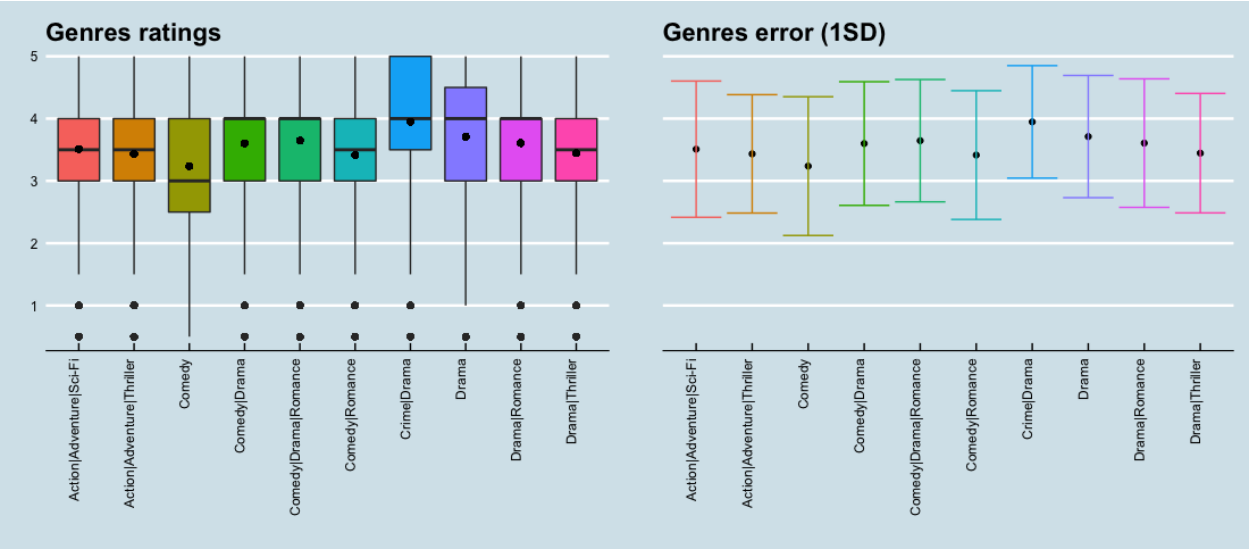
Now the RMSE is 0.8652126. The values less 0.5 and greater than 5 were brought into the range.

2.3 Genre effect

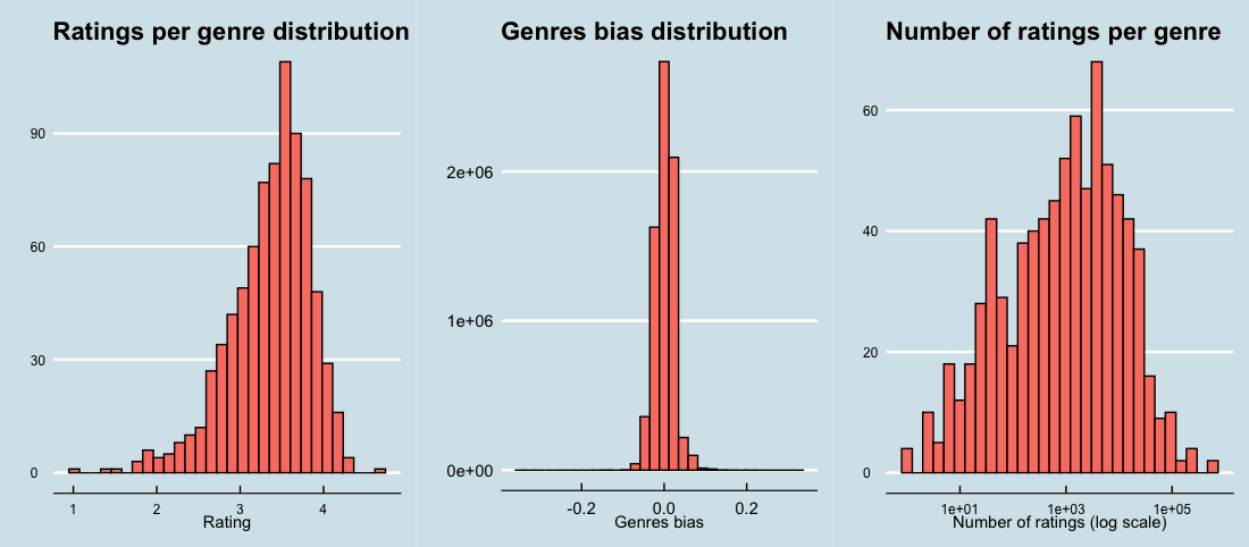
At edx we have 20 classifications for genres and some are much more evaluated than others. Drama and Comedy are rated more than 30 times compared to Documentary, for example.



Most movies do not have one, but some of those principal genres. Thinking of each combination as a specific genre we now have 797 categories. Let's look at the most popular genres to better understand how the data is distributed.



A distribution of genres ratings analysis, show us the values are accumulated around the average. We also notice that, after subtracting movie and user bias, the genre bias scope is very narrow.



There are genres with only one evaluation and others with more than 500 thousand grades, so we must use regularization.

$$b_g = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u)$$

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

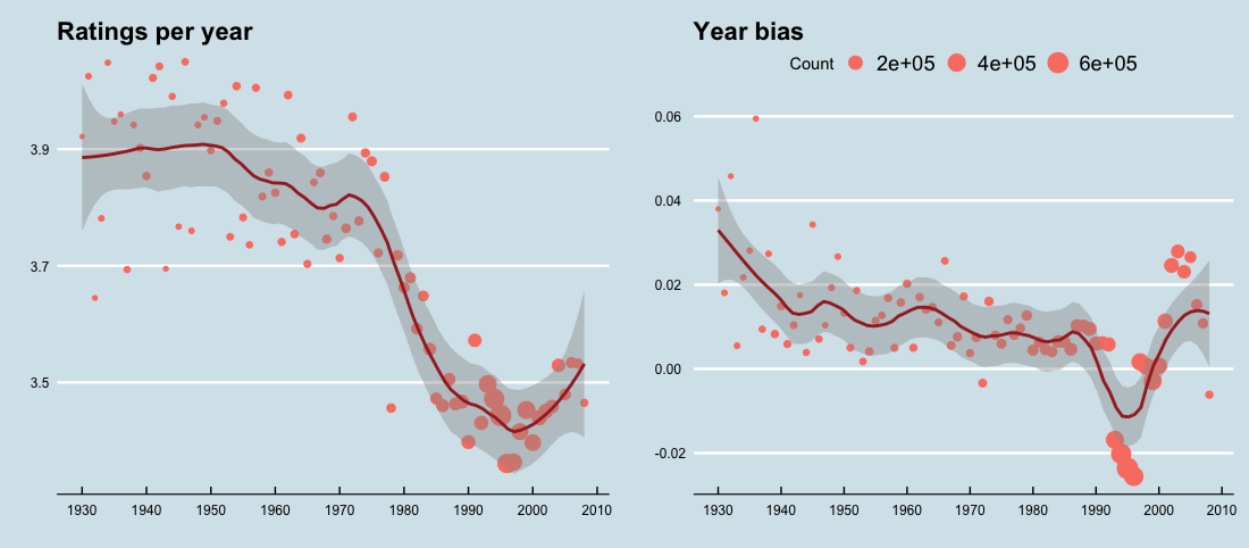
Now we have reached an RMSE equal to 0.8648738, which is already lower than the target, 0.8649.

2.4 Time effect

Next to the movie's title, in edx, we have the release year. This data was separated into a new column and thus we gained another variable called "year". From the variable "timestamp", which represents the date of the evaluation with hours, minutes and seconds, only the year was separated into a new variable "r_date".

2.4.1 Release date

We can see a strong influence of the release year on the movie grades. It is clear the older the movie, higher its score tends to be. But why? Movies released between 1930 and 1970 are classics, have already undergone the filter of time and if they are still popular after, it is because they are probably good. In the 80's it is possible to notice a sharp drop in grades, having the worst phase in the 90's, which is an important decade because they are the most evaluated movies. Analyzing the release year bias, we note that the effect is softened after the other biases have been removed, but still relevant, especially in the 90's where we noticed a deep depression. The most recent releases have slightly better ratings and it makes an important difference for year bias.



2.4.1.1 Local weighted regression (loess)

To smooth the line over the years, local weighted regression (loess) was used, which allows us to consider different sizes of data windows using Taylor's theorem, which tells us that if you look closely at any smooth function $f(x)$, it will look like a line. We can consider larger window sizes with the linear assumption than with a constant. For each point $x0$ loess defines a window and fits a line within that window (h).

$$E[Y_i \mid X_i = x_i] = \beta_0 + \beta_1(x_i - x_0) \quad if \quad \mid x_i - x_0 \mid \leq h$$

The value adjusted at $x0$ becomes our estimate $\hat{f}(x0)$. The final result is a smoother fit, as we use larger sample sizes to estimate our local parameters. Loess maintains the same number of points used in the local adjustment, which is controlled by the span argument that expects the proportion of the total data. Instead of using least squares, we minimize a weighted version:

$$\sum_{i=1}^{N} w_0(x_i)[Y_i - \{\beta_0 + \beta_1(x_i - x_0)\}]^2$$

Instead of the Gaussian kernel, loess uses a function called Tukey's triple weight that reaches the values closer to the maximum.

$$W(u) = \left(1 - \mid u \mid^3\right)^3 \quad if \quad \mid u \mid \leq 1 \quad and \quad W(u) = 0 \quad if \quad \mid u \mid > 1$$

$$w_0(x_i) = W\left(\frac{x_i - x_0}{h}\right)$$

Taylor's theorem tells us if we don't look so closely at the mathematical function, it looks like a parable. This is the standard function procedure and can be adjusted through the degree parameter, being 1 to adjust polynomials of degree 1 (lines) and 2 for parabolas. To smooth the estimate of the launch year bias (b_y), degree 1 and span 0.05 were used, chosen through cross validation between train set and test set. As released movies number per year varies a lot over decades, we will use regularization.

$$b_y = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u - b_g)$$

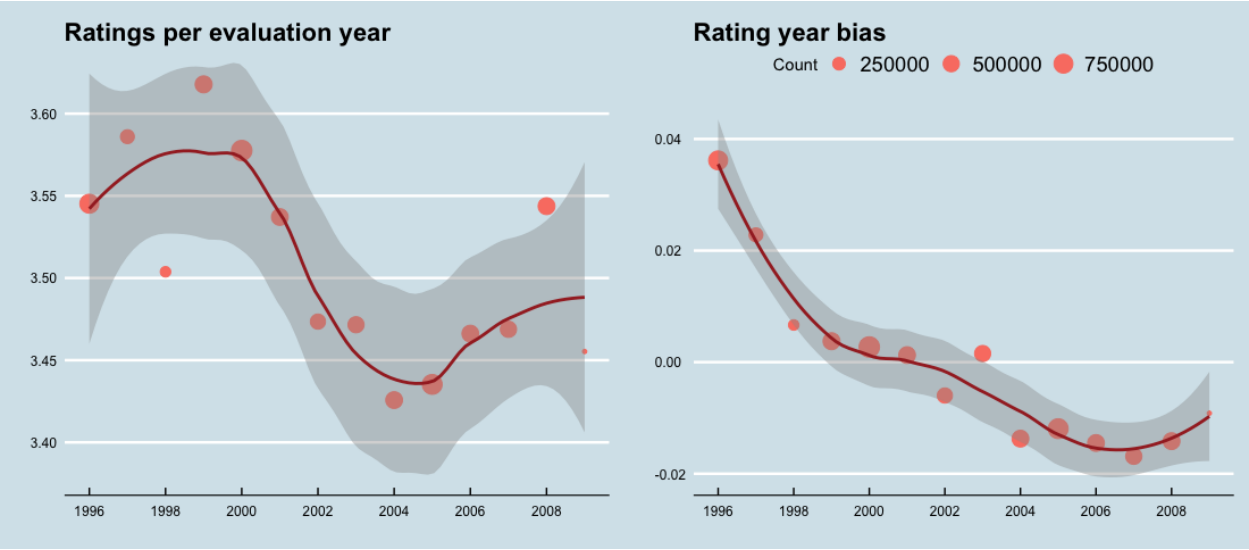
$$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + \varepsilon_{u,i}, \ with \ f(b_y) \ as \ loess \ function \ of \ release \ movie \ year.$$

```
## Call:
## loess(formula = b_y ~ year, data = b_y, span = 0.05, degree = 1,
##     family = "symmetric")
##
## Number of Observations: 94
## Equivalent Number of Parameters: 35.65
## Residual Scale Estimate: 0.009131
## Trace of smoother matrix: 41.08  (exact)
##
## Control settings:
##   span     :  0.05
##   degree   :  1
##   family   :  symmetric      iterations = 4
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

Now we reach the RMSE of 0.8647399.

2.4.2 Rating date

Looking at the grades given over the years, we can see a change in the level between 2000 and 2005. Perhaps users have become more critical over time. And it seems that even after removing the biases of the movie, user, genre and release year, the evaluation year bias shows an even clearer relationship with the grade.



The number of evaluations per year is a little more constant, but even so, we notice years with far fewer evaluations than others, so we will continue with regularization.

$$b_r = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u - b_g - f(b_y))$$

As with release year, local weighted regression was used to smooth the curve, but this time with span = 0.3.

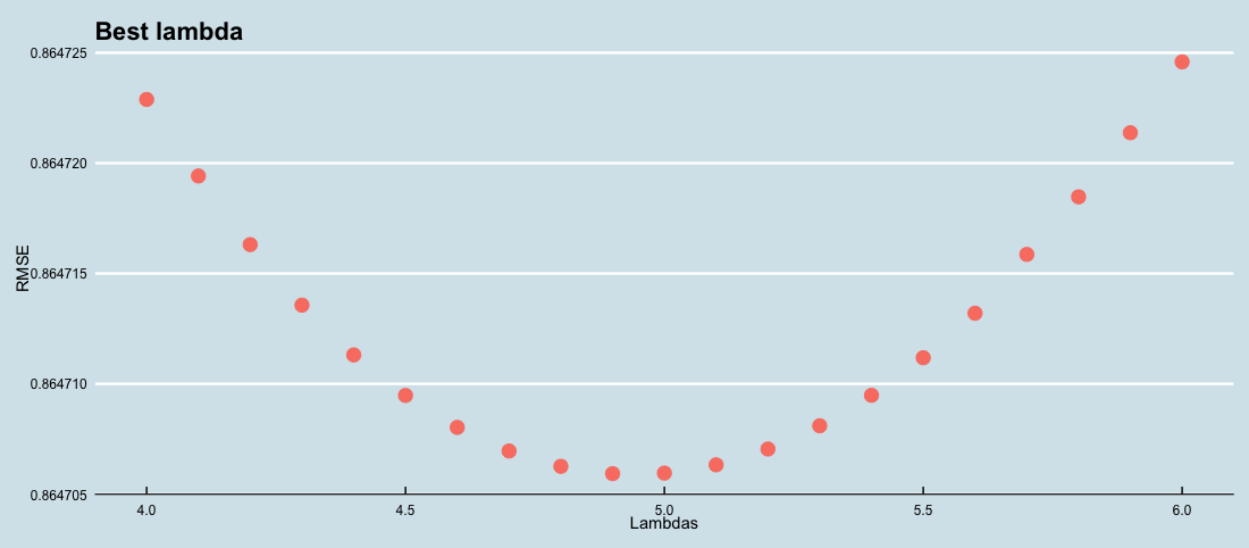
$$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + \varepsilon_{u,i}, \ \textit{with } f(b_r) \textit{ as loess function of evaluation year.}$$

```
## Call:
## loess(formula = b_r ~ r_date, data = b_r, span = 0.3, degree = 1,
##     family = "symmetric")
##
## Number of Observations: 14
## Equivalent Number of Parameters: 5.8
## Residual Scale Estimate: 0.001381
## Trace of smoother matrix: 6.89  (exact)
##
## Control settings:
##   span     :  0.3
##   degree   :  1
##   family   :  symmetric      iterations = 4
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

Now we reach the RMSE of 0.8645866.

2.5 Best lambda (λ)

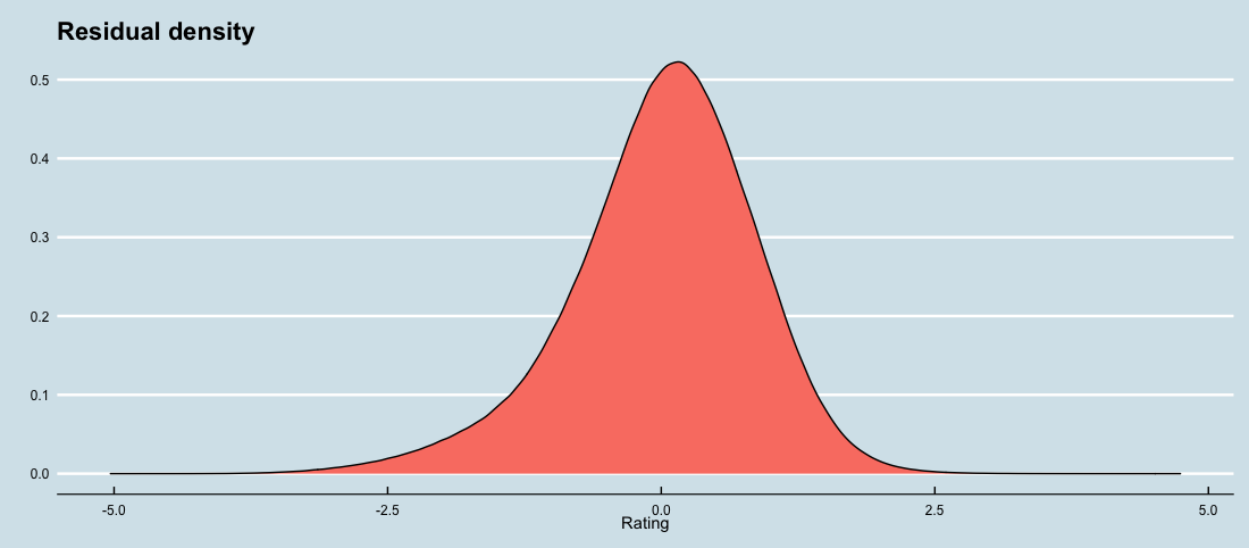
Lambda is our adjustment parameter for the number of observations in all bias filters and cross validation between train set and test set was used to define it.



The value that minimized the RMSE was 4.9.

2.6 Residual prediction

After subtracting all biases, we still have residues and can see these residuals have a zero average and a normal distribution.



Understanding these residuals are correlated, such as belonging to the same movies series or starring the same actor, linear, logistic and weighted local regressions were used.

2.6.1 Relevant data set

To use movie genres as a variable, the column gen_ind was inserted, a numerical index of genres combinations.

As it is a large dataset, to run these regressions we have problems of time and RAM. The solution was to reduce the database while keeping the most relevant data. To this end, we exclude users who rated less movies than the average, as well as movies that received less grades than average. We now have just over 4 million rows. It is still big data, but it is already possible to run simpler regressions.

2.6.2 Linear regression

For each standard deviation, σX, that x increases above the average μX, Y increases ρ standard deviation σY above the average μY with ρ the correlation between X and Y. The formula for the regression is therefore:

$$\left(\frac{Y-\mu_Y}{\sigma_Y}\right) = \rho\left(\frac{x-\mu_X}{\sigma_X}\right)$$

$$Y = \mu_Y + \rho\left(\frac{x-\mu_X}{\sigma_X}\right)\sigma_Y$$

If the correlation is positive and less than 1, our forecast is closer, in standard units, to the average than the value used to predict, x, is the average of xs. To add regression lines to the graphs, we will need the formula above in the format:

$$y = b + mx \text{ with slope } m = \rho\frac{\sigma_y}{\sigma_x} \text{ and intercept } b = \mu_y - m\mu_x$$

$$res = Y_{u,i} - \mu - b_i - b_u - b_g - f(b_y) - f(b_r)$$

Where res is the redidual for user u, movie i and their genre, release and rating year.

$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + f(res) + \varepsilon_{u,i}$ *with* $f(res)$ *as linear regression function of residuals.*

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7727 -0.4756  0.0676  0.5608  4.7469
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.199e+00  2.698e-01  15.565  < 2e-16 ***
## movieId      1.331e-06  6.271e-08  21.220  < 2e-16 ***
## userId       5.270e-08  2.033e-08   2.592  0.00954 **
## gen_ind     -3.134e-06  1.847e-06  -1.697  0.08974 .
## year        -4.676e-04  3.303e-05 -14.157  < 2e-16 ***
## r_date      -1.637e-03  1.338e-04 -12.237  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8395 on 4005694 degrees of freedom
## Multiple R-squared:  0.0001554,  Adjusted R-squared:  0.0001541
## F-statistic: 124.5 on 5 and 4005694 DF,  p-value: < 2.2e-16
```

With this model we reached the RMSE of 0.8645834.

2.6.3 Logistic regression

The logistical transformation for a proportion or rate p is defined as: $g(p) = log(p/(1-p))$ When p is a ratio or probability, the quantity being logged, $p/(1-p)$, is called probabilities. The transformation of the log makes this symmetrical. If the rates are the same, then the probability of logging is 0. Increases or decreases of times become positive and negative increments, respectively. Using the log, these fold changes become constant increases. Logistic regression is a specific case of a set of generalized linear models. The function $\beta_0 + \beta_1 x$ can take on any value, including negatives and values greater than 1. But we are estimating a probability: $Pr(Y = 1 \mid X = x)$ which is restricted between 0 and 1. The idea of generalized linear models (GLM) is: 1) to define a Y distribution that is consistent with your possible results and, 2) find a function g so that $g(Pr(Y = 1 \mid X = x))$ can be modeled as a linear combination of predictors. Logistic regression is the most commonly used GLM. It is an extension of linear regression that ensures that the estimate of $Pr(Y = 1 \mid X = x)$ is between 0 and 1. This logistic transformation converts the probability into a log of chances. A good feature of this transformation is that it converts the probabilities to symmetric around 0.

$$g\{p(x_1, x_2, ..., x_5)\} = g\{Pr(Y = 1 \mid X_1 = x_1, X_2 = x_2, ..., X_5 = x_5)\} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_5 x_5.$$

$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + f(res) + \varepsilon_{u,i}$ *with* $f(res)$ *as logistic regression function of residuals.*

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.7727  -0.4756   0.0676   0.5608   4.7469
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.199e+00  2.698e-01  15.565  < 2e-16 ***
## movieId      1.331e-06  6.271e-08  21.220  < 2e-16 ***
## userId       5.270e-08  2.033e-08   2.592  0.00954 **
## gen_ind     -3.134e-06  1.847e-06  -1.697  0.08974 .
## year        -4.676e-04  3.303e-05 -14.157  < 2e-16 ***
## r_date      -1.637e-03  1.338e-04 -12.237  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.7047648)
##
##     Null deviance: 2823511  on 4005699  degrees of freedom
## Residual deviance: 2823072  on 4005694  degrees of freedom
## AIC: 9966133
##
## Number of Fisher Scoring iterations: 2
```

With this model we reached the RMSE of 0.8645834, exactly the same as hitted with linear regression. This is expected since, like regression, the decision rule is a line, so x_2 is a linear function of x_1, what implies that logistic regression does not capture the non-linear true $p(x_1, ..., x_5)$.

2.6.4 Local weighted regression

The loess function has been explained previously and with it our prediction was:

$$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + f(res) + \varepsilon_{u,i} \quad \textit{with } f(res) \textit{ as loess function of residuals.}$$

```
## Anova for Nonparametric Effects
##                                     Npar Df Npar F     Pr(F)
## (Intercept)
## lo(r_date, span = 0.5, degree = 1)      2.6  37.51 < 2.2e-16 ***
## lo(year, span = 0.5, degree = 1)        2.4  45.12 < 2.2e-16 ***
## lo(gen_ind, span = 0.5, degree = 1)     2.3  35.20 < 2.2e-16 ***
## lo(movieId, span = 0.5, degree = 1)     3.8 317.78 < 2.2e-16 ***
## lo(userId, span = 0.5, degree = 1)      2.3   4.91  0.005225 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Anova for Parametric Effects
##                                          Df  Sum Sq Mean Sq  F value    Pr(>F)
## lo(r_date, span = 0.5, degree = 1)        1     154  154.43 219.2149 < 2.2e-16
## lo(year, span = 0.5, degree = 1)          1      91   90.74 128.7963 < 2.2e-16
## lo(gen_ind, span = 0.5, degree = 1)       1       0    0.47   0.6703  0.412938
## lo(movieId, span = 0.5, degree = 1)       1     572  572.12 812.0984 < 2.2e-16
## lo(userId, span = 0.5, degree = 1)        1       5    4.80   6.8171  0.009029
## Residuals                           4005681 2821966    0.70
##
## lo(r_date, span = 0.5, degree = 1)  ***
## lo(year, span = 0.5, degree = 1)    ***
## lo(gen_ind, span = 0.5, degree = 1)
## lo(movieId, span = 0.5, degree = 1) ***
## lo(userId, span = 0.5, degree = 1)  **
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, the default settings for degree and span were used and the RMSE achieved was 0.8644825.

2.7 Ensemble

For our final prediction we use the average of the previous predictions. The basis for all of them is the same, changing only the forecast of the residual calculated through linear, logistic and local weighted regression.

The RMSE target less than 0.8649 was achieved by closing at 0.8645046.

3 Results

The same techniques were used in edx with configurations optimized by cross validation between train set and test set. The prediction was tested with validation set and for:

$$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + \varepsilon_{u,i}$$

We achieved an RMSE equal to 0.8640852, better than all predictions between train and test set.

Using linear regression to predict the residual,

$$Y_{u,i} = \mu + b_i + b_u + b_g + f(b_y) + f(b_r) + f(res) + \varepsilon_{u,i}$$

we obtained an RMSE equal to 0.8640676 just a little better.

With logistic regression the same result remains, RMSE equal to 0.8640676. And finally with local weighted regression we got the best result, RMSE equal to 0.8639477.

The final prediction, found through the three previous predictions average, reached RMSE equal to 0.8639821, which satisfactorily reached the project objective and proved to be an efficient algorithm that is not time consuming, but takes a few hours anyway.

Methods	RMSE
Effects	0.8640852
Effects + LM	0.8640676
Effects + GLM	0.8640676
Effects + LOESS	0.8639477
Ensemble	0.8639821

4 Conclusion

We can say this project was successful in developing a recommendation system algorithm to predict movie ratings for the 10m version of MovieLens data using the training and validation sets provided. Biases were identified and used, regression models and ensemble methods were trained.

The performance of the model's evaluation using the RMSE (Root Mean Square Error) showed linear regression model with regularized effects on users, movies and genres added to the smoothed effects of rating and release years, is efficient and satisfies project's objective achieving RMSE equal to 0.8640852, less than 0.8649. Using regressions to predict the residual in a reduced data set, local weighted was the most efficient model reaching RMSE equal to 0.8639477.

Other different machine learning models may also improve results even more, but hardware limitations, such as RAM and processing time, were a constraint.

For future model development, using dimensionality reduction techniques as matrix factorization with stochastic gradient descending, like PCA and SVD, can help overcome these problems and then allow the use of more robust models with random forest, k-nearest neighbors and many others. These models will be based on the proximity between users and movies through correlated behavior, finding the vectors p and q that permit us to rewrite the matrix r with m rows and n columns with a reduced fraction of information.