Movielens project

Lucas Nobre

4 de janeiro de 2019

Movielens project

This is a project about predicting user's movies ratings to create better recommendations to them, by the end of it we hope to create a good way to predict a user rating and tell him movies he might like.
As seen in the HarvardX's course: Data Science: Machine Learning, we will be using a 10M version of the movielens dataset. The data is then separated as test and validation sets using a given script.

About the dataset

Now let's take a look at some of the dataset characteristics. We will be looking at the edx dataset, since the validation dataset is just for testing our model. Then, for the edx dataset we want to answer some questions:

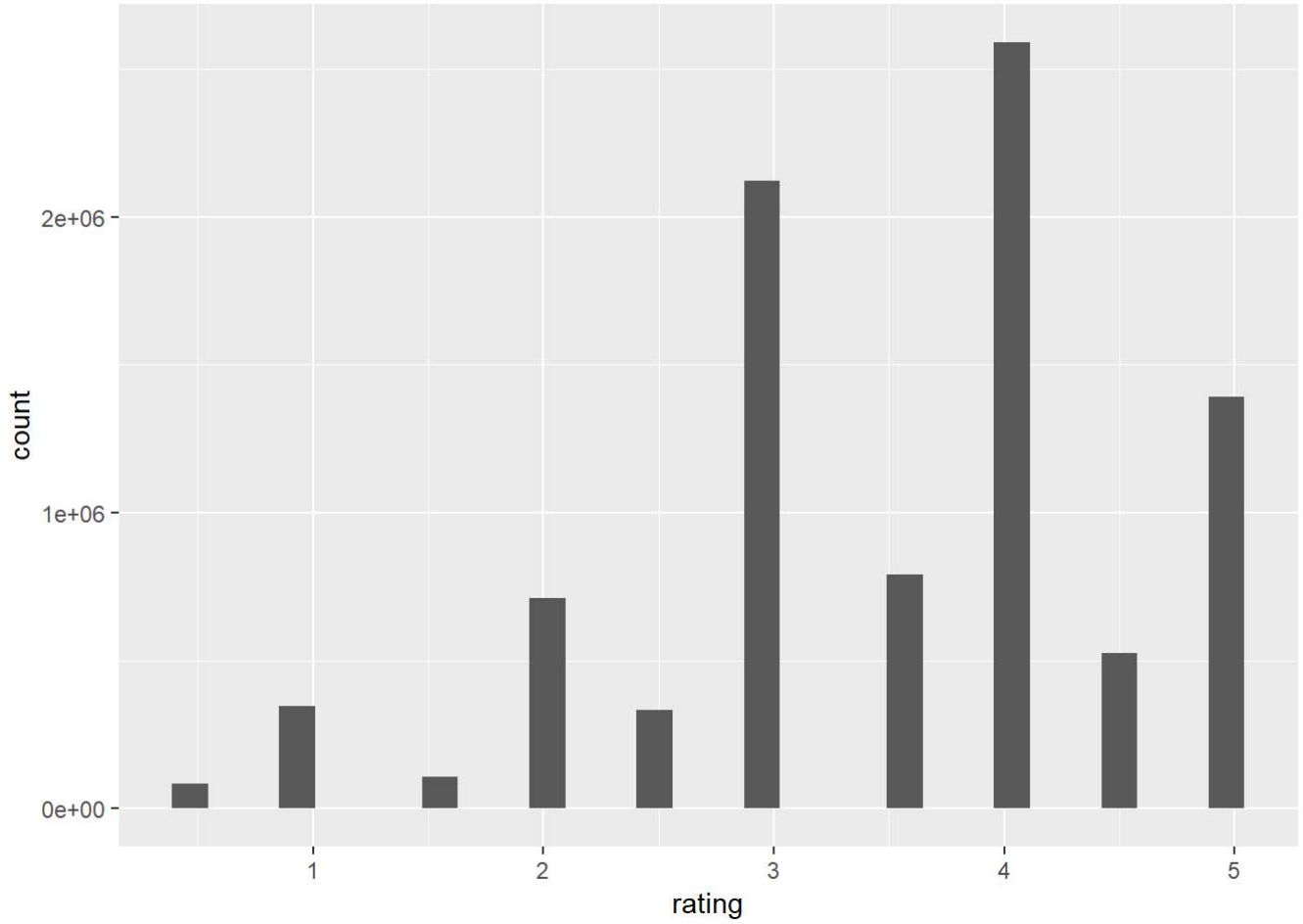
- How big is our dataset?
- What are the values of our ratings? Are they rounded numbers? Do we have 0's?
- How many users we have? And movies?

This table has the answers we need.

Rows	Columns	Users	Movies	Ratings
9000055	6	69878	10677	5, 3, 2, 4, 4.5, 3.5, 1, 1.5, 2.5, 0.5

We can see the dataset is very big, with many users and different movies. Now, about the ratings what are the most frequent ratings? This is a good question because it will boost our accuracy later, so let's check the ratings

```
edx %>% group_by(rating) %>%  ggplot(aes(rating)) + geom_histogram()
```



There are more .0 numbers than .5, this is really important, because we will be rounding our ratings to .0.

Building our model

First of all, we will split the dataset into two partitions, the training set and the testing set. After this we will get the average rating for all movies

```
mu = mean(edx$rating)
```

So, the most simple model we can build, is to look for an average rating. Our average will be 3.5124652. Now, for checking our model we will be using RMSE and accuracy. For RMSE we will create our own function:

```
my_rmse <- function(prediction, true){
  sqrt( mean( (prediction - true)^2 , na.rm = TRUE ) )
}
```

Now let's check our first model! First we will check the RMSE.
RMSE = 1.0603313
And now for the accuracy:

```
rounded_prediction <- round(mu/0.5)*0.5
accuracy <- mean(rounded_prediction == validation$rating)
```

Our accuracy is: 0.0881401
Notice that we are rounding the prediction to it's nearest possible rating. Both the RMSE and the accuracy are bad, but we can make this accuracy a little higher. Notice that our most common rating is 4, and more then doubles the amount of 3.5 rating. So why are we using 3.5 to test our accuracy? Let's round our prediction to 4 and test the accuracy again.

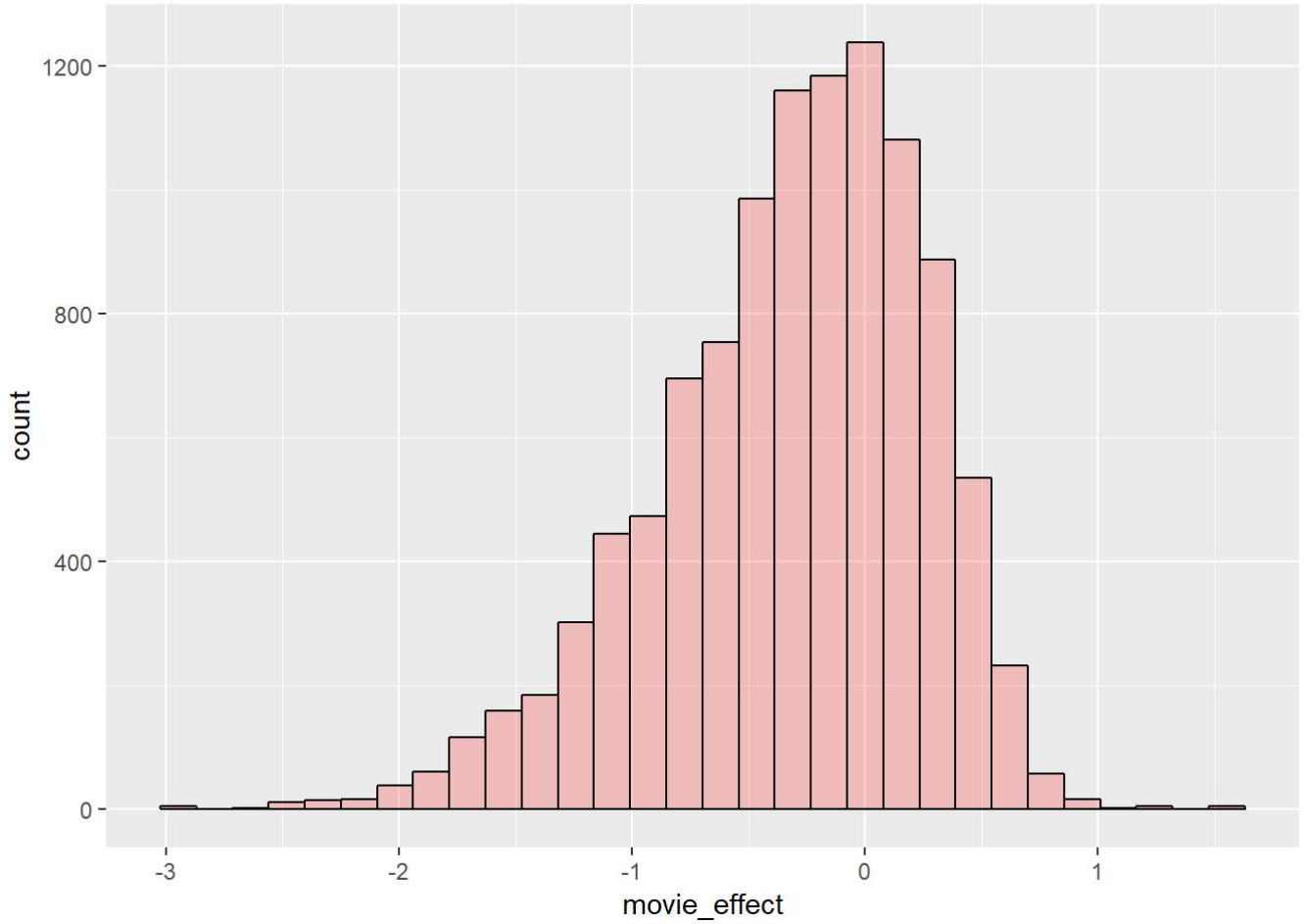
```
new_rounded_prediction <- round(mu)
new_accuracy <- mean(new_rounded_prediction == validation$rating, na.rm = TRUE)
```

Our new accuracy is: 0.2874203 This is a better number. Let's improve more!

Checking the movie effect

Our first prediction was really bad. A RMSE higher than 1 is at least one star wrong. This happens because we are only using the average value from the movies. But every movie is different from one another, so their average ratings will not be the same. This means a movie will have an effect on its own rating, and this will be called the movie effect. To calculate the movie effect we will remove the mean from the movie's mean.

```
movies_ratings <- edx %>% group_by(movieId) %>%
  summarize(movie_effect = mean(rating - mu))
movies_ratings %>% ggplot(aes(movie_effect))  + geom_histogram(color = 'black', fill = 'red',
 alpha = .2)
```



It is not centered at 0 as we first expect. This is because the most viewed movies have a high rating average, as we can see in this table.

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

title <chr>
Pulp Fiction (1994)
Forrest Gump (1994)

title <chr>
Silence of the Lambs, The (1991)
Jurassic Park (1993)
Shawshank Redemption, The (1994)
Braveheart (1995)
Fugitive, The (1993)
Terminator 2: Judgment Day (1991)
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
Apollo 13 (1995)

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

Now let's make some predictions and see how our model performs with the movie effect.

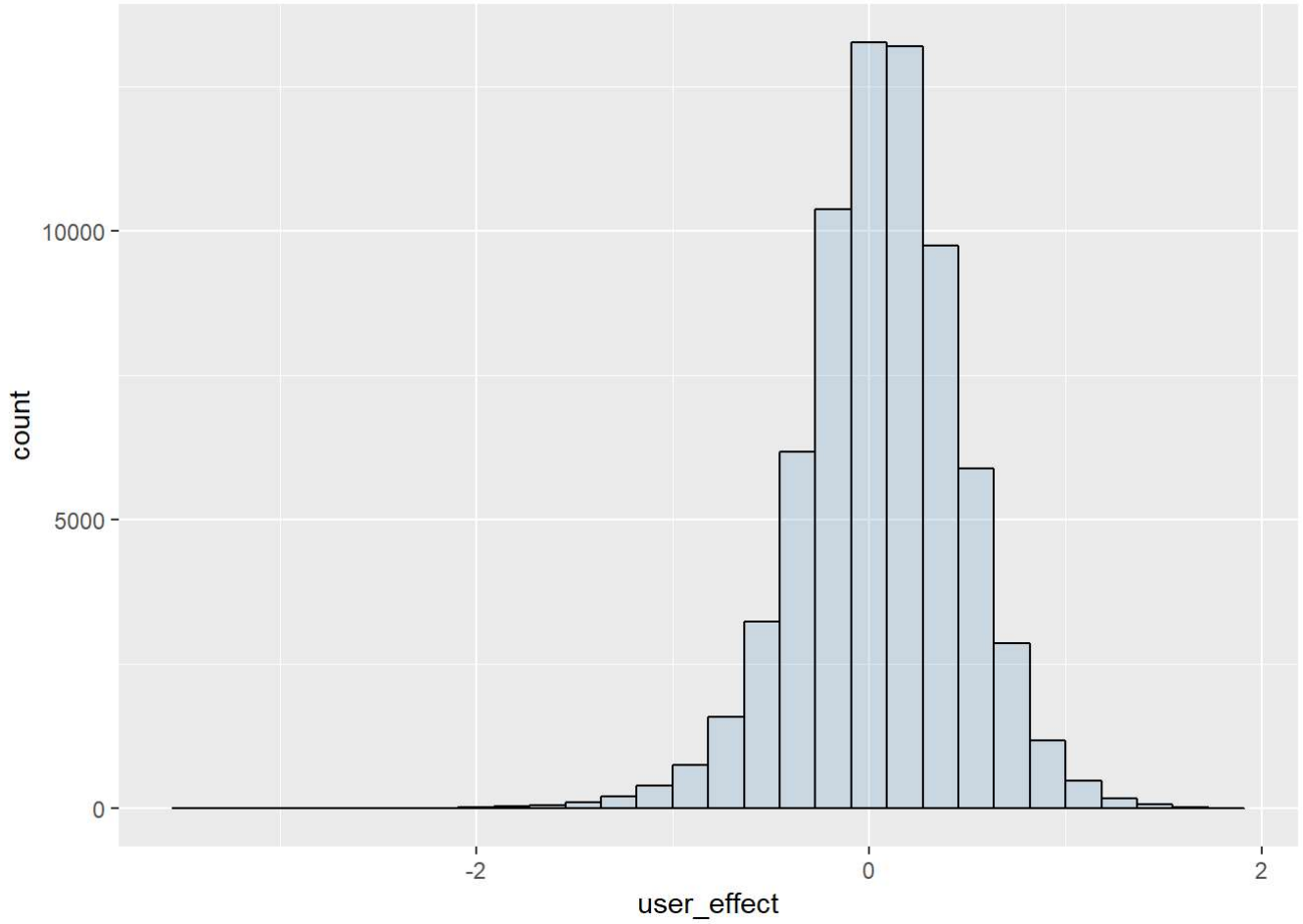
```
movie_prediction <- mu + validation %>%
  left_join(movies_ratings, by='movieId') %>% .$movie_effect
movie_rmse <- my_rmse(movie_prediction, validation$rating)
movie_accuracy <- mean(round(movie_prediction) == validation$rating, na.rm = TRUE)
```

Our new RMSE with the movie effect is: 0.9439087
and our accuracy with the movie effect is: 0.3260703
This is a great improvement. Our RMSE is already lower than 1, but not that much. Our next step will be checking for the user effect.

Looking at user effect

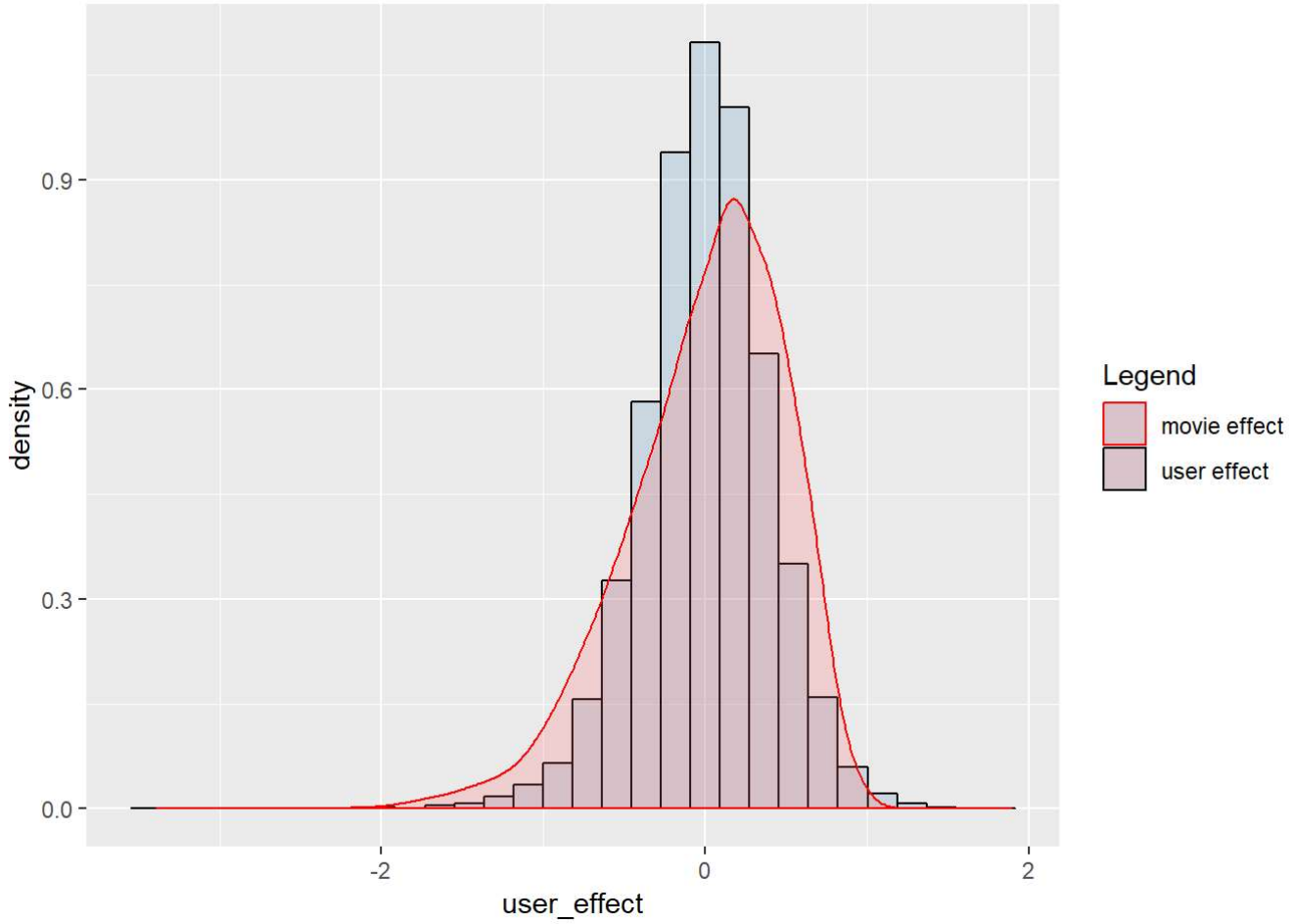
We are trying to build a recommendation system for the users, so understanding how a given user rates movies is a good approach. The user effect will be calculated like the movie effect, we will get each user rating and remove the average rating and movie effect of the movie. Let's see how the users like to rate movies.

```
user_ratings <- edx %>% left_join(movies_ratings, by = 'movieId') %>%
  group_by(userId) %>% summarize(user_effect = mean(rating - mu - movie_effect))
user_ratings %>% ggplot(aes(user_effect))  + geom_histogram(color = 'black', fill = 'steelblu
e', alpha = .2)
```



Now this graph is more centered at 0 and it looks more like a normal curve then the movie effect graph. Let's make a plot that shows the difference between the two of them.

```
edx %>% left_join(movies_ratings, by = 'movieId') %>%
  left_join(user_ratings, by = 'userId') %>%
  ggplot() + geom_histogram(aes(user_effect, y=..density..,colour = 'user effect'), fill='ste
elblue', alpha = .2) + geom_density(aes(movie_effect, colour = 'movie effect'), alpha = .2, f
ill = '#FF6666', bw = 0.1) + scale_colour_manual("Legend", values = c("red", "black"))
```



The movie average has a left tail and the user average is more centered and distributed. This means it is more usual that a movie is below average than a user that is more strict with their ratings. This also means that userId is a random variable.

```
user_predictions <- validation %>%
  left_join(movies_ratings, by = 'movieId') %>%
  left_join(user_ratings, by = 'userId') %>%
  mutate(means = mu + user_effect + movie_effect,
         pred = ifelse(means > 5, 5, ifelse(means < 1, 1, means)))

user_rmse <- my_rmse(user_predictions$pred, validation$rating)
user_accuracy <- mean(round(user_predictions$pred) == validation$rating, na.rm = TRUE)
```

Our RMSE is now: 0.8651351.
And our accuracy: 0.3615404.
This is a good improvement.

Improving our model

Regularization

We are calculating the effects without minding how many votes each user or movie has, this can lead to a wrong rating to a movie, new movies don't have many votes, so their average rating is probably a bad estimation. We can see that the best rating averages are from movies with less than 5 ratings.

```
edx %>% group_by(movieId) %>% summarize(avg = mean(rating), n = n()) %>% arrange(desc(avg))
```

movieId <dbl>	avg <dbl>	n <int>
3226	5.000000	1
33264	5.000000	2
42783	5.000000	1
51209	5.000000	1
53355	5.000000	1
64275	5.000000	1
5194	4.750000	4
26048	4.750000	4
26073	4.750000	4
65001	4.750000	2

1-10 of 10,000 rows Previous **1** 2 3 4 5 6 … 1000 Next

To handle this problem we will perform a regularization on the ratings, dividing the number of ratings for a constant number, so the average ratings with low number of votes have a lower weight. To perform this regularization we will choose a number lambda that maximize our RMSE

```
lambdas = seq(0, 10, 0.5)

reg_preds <- sapply(lambdas, function(lambda){

  movie_reg_avgs <- edx %>%
    group_by(movieId) %>%
    summarize(movie_effect = sum(rating - mu)/(n() + lambda))

  user_reg_avgs <- edx %>% left_join(movie_reg_avgs, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(user_effect = sum(rating - mu - movie_effect)/(n() + lambda))

  reg_ratings <- validation %>% left_join(movie_reg_avgs, by = 'movieId') %>%
    left_join(user_reg_avgs, by = 'userId') %>%
    mutate(means = mu + movie_effect + user_effect,
           pred = ifelse(means > 5, 5,
                         ifelse(means < 1, 1, means))) %>%
    .$pred
})
colnames(reg_preds) <- lambdas
rmses <- apply(reg_preds, 2, my_rmse, validation$rating)
best_lambda <- lambdas[which.min(rmses)]
best_reg <- reg_preds[, best_lambda]
reg_rmse <- my_rmse(best_reg, validation$rating)
reg_accuracy <- mean(round(best_reg) == validation$rating, na.rm = TRUE)
```

The best lambda we can choose is: 5 and our new RMSE is: 0.8648156 Not a big improvement, but it is a improvement.

Creating the submission file

```
# Adding ratings predictions to validation table

validation <- validation %>% select(-rating)
validation$rating <- best_reg
# Ratings will go into the CSV submission file below:

write.csv(validation %>% select(userId, movieId, rating),
          "submission.csv", na = "", row.names=FALSE)
```

Conclusions

Calculating the movie_effect and user_effect were efficient ways to lower the RMSE to a good value, validating our methodology and making good recommendation of movies to the users. Using the userId 8, we can give him good movie recommendations now!

```
validation %>% filter(userId == 8) %>% arrange(desc(rating))
```

userId <int>	movieId <dbl>	timestamp <int>	title <chr>
8	678	1111622859	Some Folks Call It a Sling Blade (1993)
8	1253	1115860094	Day the Earth Stood Still, The (1951)
8	1210	1115858270	Star Wars: Episode VI - Return of the Jedi (1983)
8	7090	1111624053	Hero (Ying xiong) (2002)
8	6539	1111624023	Pirates of the Caribbean: The Curse of the Black Pearl (2003)
8	32	1111623496	12 Monkeys (Twelve Monkeys) (1995)
8	1307	1116548923	When Harry Met Sally... (1989)
8	1265	1116548937	Groundhog Day (1993)
8	4978	1115859007	Lantana (2001)
8	6807	1115858412	Monty Python's The Meaning of Life (1983)

1-10 of 73 rows | 1-4 of 6 columns Previous **1** 2 3 4 5 6 … 8 Next

We had lots of improvement creating this model, we can see it in this table.

Method	RMSE
Mean	1.0603313
Movie Effect	0.9439087
Movie + User Effects	0.8651351
Regularized Movie + User Effects	0.8648156