MovieLens Project

Aadyant Khatri

21/01/2022

Overview

This project is aimed at predicting the rating a user would give to a movie. This would help in recommending relevant movies to users.

For this project, **MovieLens** data set from GroupLens Research was downloaded. This data set contains 10 million ratings given by 72,000 users to 10,000 movies.

The type of model I intended to create in this project is a part of **recommendation systems**. In this model, for each outcome, the set of predictors are different which makes such machine learning models more complicated than others.

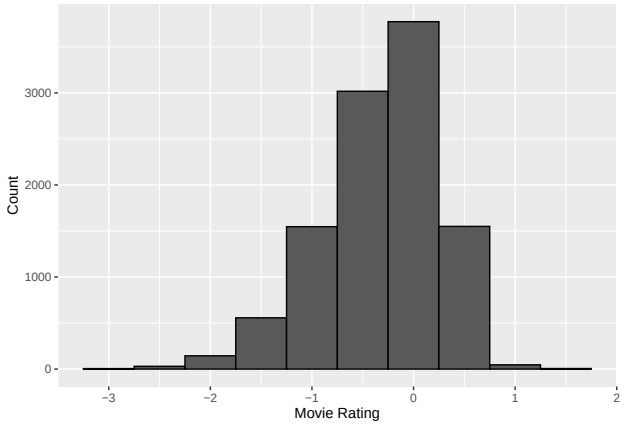
Methodology

Model - 1

It was initially assumed that every variation in the rating of different movies by different users is explained by **random errors**. In such a case, using the average rating of all movies across all users as the prediction would give the least error. This was treated as the base model and it gave Root Mean Square Error (RMSE) = 1.0608975

Model - 2

The RMSE obtained with the base model was further reduced by super-imposing the **movie effect**. This approach takes into consideration the fact that some movies are relatively better than others and hence get a higher rating.



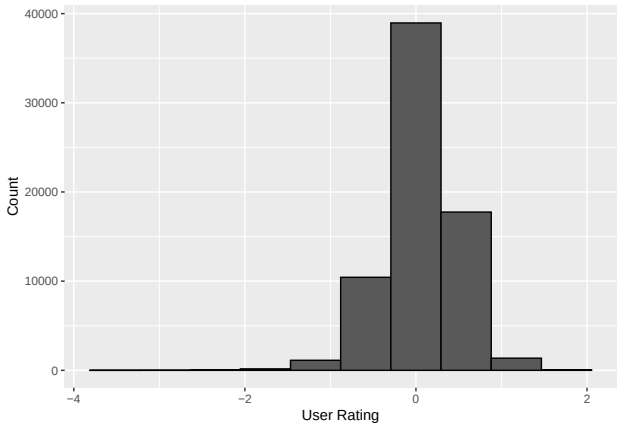
To capture this effect, for each movie, difference between its true ratings and the average rating was calculated and then its average was taken.

```
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(movie_avg = mean(rating - avg_rating))
```

This method gave RMSE = 0.9405469

Model - 3

The model was improved by also considering the **user effect** which takes into account the fact that different users prefer different genres of movies.



This effect was approximated by subtracting the average rating and movie effect from the true ratings and then averaging the values for each user.

```
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(user_avg = mean(rating - avg_rating - movie_avg))
```

Model 3 resulted in RMSE = 0.8629016

Model - 4

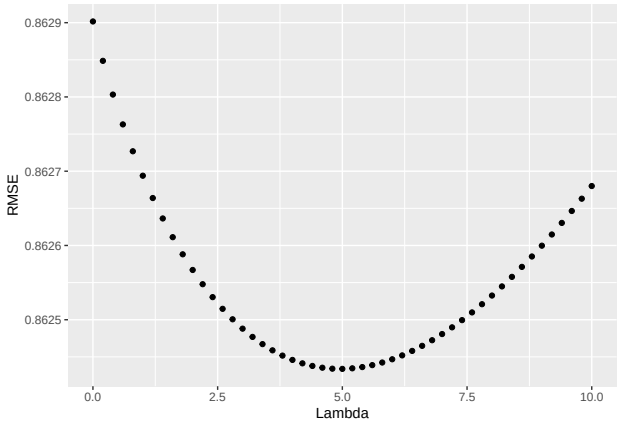
To refine the model further, I used **regularization**. Regularization constrains the total variability of the effect sizes by penalizing large estimates that come from small sample sizes. This is done by dividing the effects by a parameter called lambda.

```
# trying tuning parameter value = 3
lambda <- 3

# Regularization on movie specific effects
regularized_movie_avgs <- train_set %>%
  group_by(movieId) %>%
```

```
  summarize(regularized_movie_avg = sum(rating - avg_rating)/(n() + lambda))

# Regularization on user specific effects
regularized_user_avgs <- train_set %>%
  left_join(regularized_movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(regularized_user_avg = sum(rating - regularized_movie_avg - avg_rating)/(n() + lambda))
```

Here, lambda is a hyper-parameter and after testing out multiple values, its optimum value was found and then used for making the final prediction.

```
# Testing multiple lambda values
lambdas <- seq(0, 10, 0.2)

RMSEs <- sapply(lambdas, function(l){

  regularized_movie_avgs <- train_set %>%
    group_by(movieId) %>%
    summarize(regularized_movie_avg = sum(rating - avg_rating)/(n() + l))

  regularized_user_avgs <- train_set %>%
    left_join(regularized_movie_avgs, by="movieId") %>%
    group_by(userId) %>%
    summarize(regularized_user_avg = sum(rating - regularized_movie_avg - avg_rating)/(n() + l))

  model_4_predictions <- test_set %>%
    left_join(regularized_movie_avgs, by = "movieId") %>%
    left_join(regularized_user_avgs, by = "userId") %>%
    mutate(prediction = avg_rating + regularized_movie_avg + regularized_user_avg) %>% .$prediction

  return(RMSE(model_4_predictions, test_set$rating, na.rm = TRUE))
})
```



The lambda that gave the best RMSE was found to be = 5 and it yielded RMSE = 0.8624338

Results

The RMSEs obtained with each model have been compiled in the table below.

Method	RMSE
Base Model	1.0608975
Movie Effect Model	0.9405469
Movie + User Effects Model	0.8629016
Regularized Model (lambda = 3)	0.8624880
Regularized Model (using optimum lambda)	0.8624338

My last model which factored in movie and user specific effects, and also regularized the values, performed the best. This model was used to make the final prediction on the validation data.

```
final_movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(final_movie_avg = sum(rating - avg_rating)/(n() + min_lambda))

final_user_avgs <- train_set %>%
  left_join(final_movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(final_user_avg = sum(rating - final_movie_avg - avg_rating)/(n() + min_lambda))

final_predictions <- validation %>%
  left_join(final_movie_avgs, by = "movieId") %>%
  left_join(final_user_avgs, by = "userId") %>%
  mutate(prediction = avg_rating + final_movie_avg + final_user_avg) %>% .$prediction
```

This gave RMSE = **0.8648475**

Conclusion

I was able to create a model which could predict the rating of movies by different users with a decent accuracy (low RMSE).

This project helped in explaining the variability in ratings because of various effects. It also became evident that regularization can be used for reducing the error as it ensures that the effects don't explode into large values.

However, a limitation of my model is that it requires every movie and user, for which prediction has to be made, must be present in the training data set. This is because it is necessary to calculate the movie effect and the user effect for every movie and user respectively. Thus, my model may not work with new movies and users.