Movielens-Project

SusanSunny

12/31/2020

1 Introduction

1.1 Project Aim

The aim of this project is to build a movie recommendation system. The program should predict how a user will rate a certain movie based on the past ratings.

1.2 Project Overview

To complete this task several methods of Machine Learning will be used:

Firstly, linear regression is used to build the recommendation system. The variables included in the linear regression in my analysis are movie, user, genre and the year.

Secondly, regularization is used to improve the results. Small sample sizes have a higher variability and the error is likely to be larger. Regularization is a method that penalizes estimates that come from small sample sizes with the aim of reducing the error.

2 Methods and Analysis

2.1 Data exploration and visualization

Before doing an analysis, it is important to be familiar with the data set. In this chapter the data set is explored. The data set provided for this project is split into train and test set. The training set "edx" contains 6 columns (userId, movieId, rating, timestamp, title and genres) and 9000055 rows. The test set validation contains 999999 rows, around 10% compared to the number of ratings in the training set.

```
dim(edx)
```

```
## [1] 9000055       6
```

```
head(edx)
```

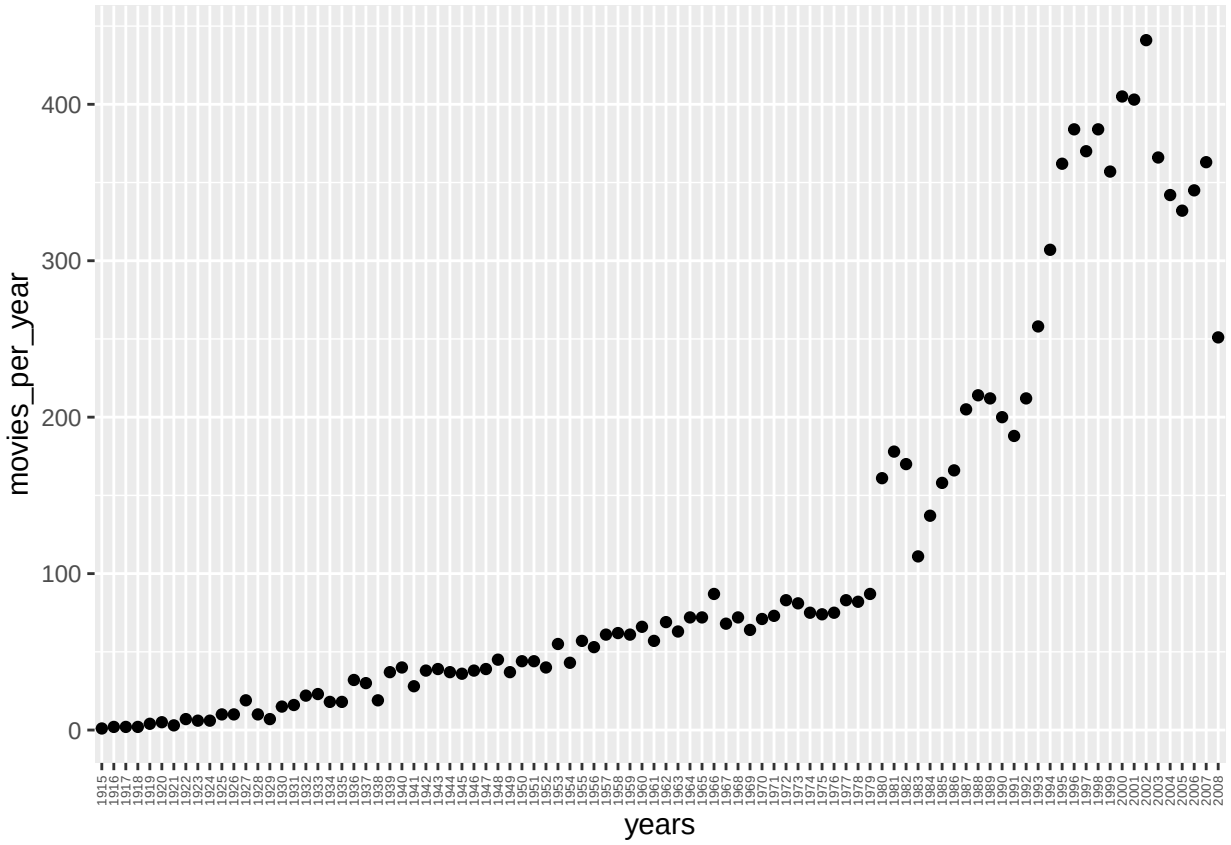
```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
```

```
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:          Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

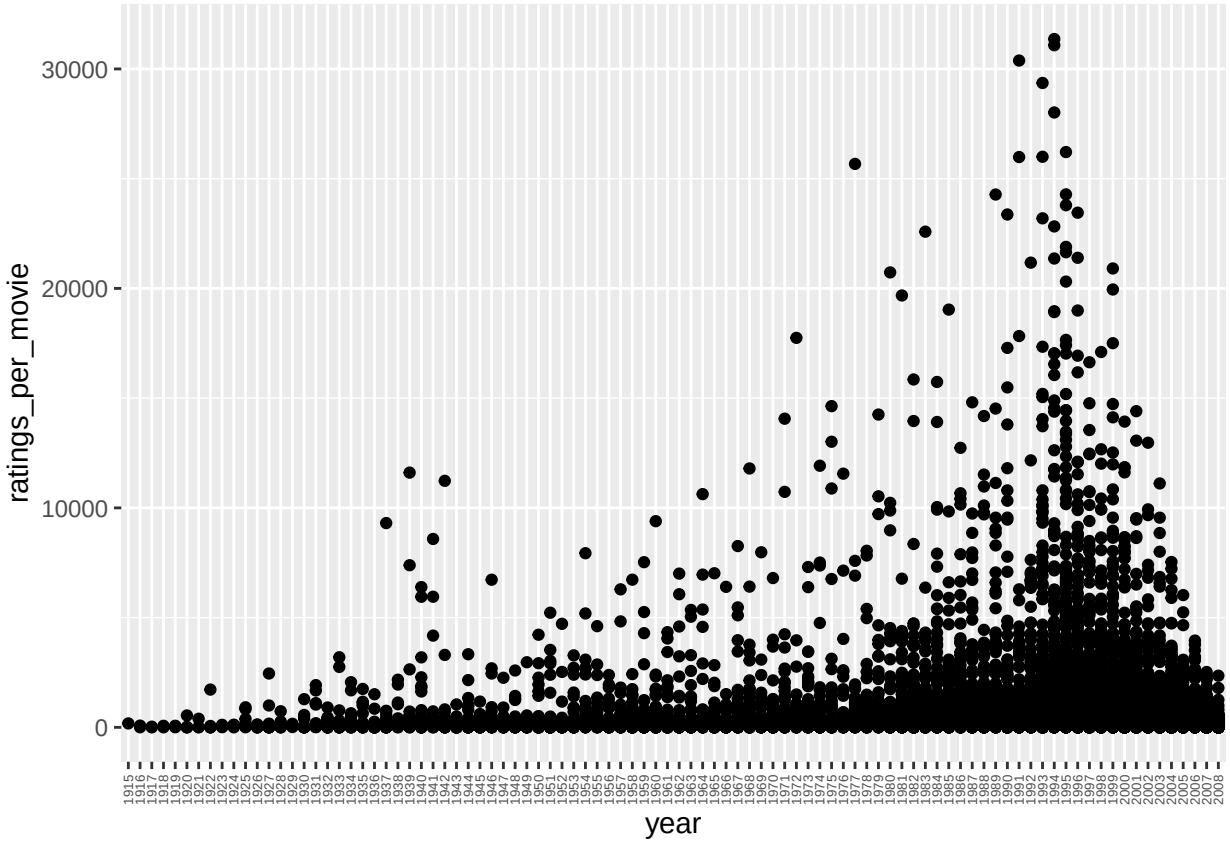
```
dim(validation)
```

```
## [1] 999999      6
```

The year the movie was released is included in the title and was extracted using regular expressions. In the following plot, the number of movies per year can be seen. The number of movies is increasing steadily until around 1978, from then the number of movies is increasing even faster.



The following plot shows the number of ratings per movie plotted against the year. Similarly to the plot above, the number of ratings per movie increases over the years until recent years. For recent movies, there are not as many ratings yet.



2.2 Linear Regression: Movie, User, Genre and Year Effect

The basic concept of Machine Learning is to split a data set into training set and test set and to only use the training set to build the model, whereas the test set should only be used to test the model. In this project, the training set is called "edx" and the test set is called "validation".

The basic method of the model is linear regression with the following equation: pred = mu + b_i + b_u + b_g + b_y. (pred = predicted rating, b_i = movie effect, b_u = user effect, b_g = genre effect, b_y = year effect)

```
#Linear Regression: Movie + User + Year + Genre Effect Model
mu <- mean(validation$rating) # overall average rating

#Movie Effect
b_i<- edx%>%
  group_by(movieId)%>%
  summarize(b_i = mean(rating-mu))

#User Effect
b_u<- edx%>%
  left_join(b_i, by = "movieId")%>%
  group_by(userId)%>%
  summarize(b_u = mean(rating-mu-b_i))

#Genre Effect
```

```
b_g<- edx%>%
  left_join(b_i, by = "movieId")%>%
  left_join(b_u, by = "userId")%>%
  group_by(genres)%>%
  summarize(b_g = mean(rating-mu-b_i-b_u))

#Year Effect (year in which the movie was released)
b_y<- edx%>%
  left_join(b_i, by = "movieId")%>%
  left_join(b_u, by = "userId")%>%
  left_join(b_g, by= "genres")%>%
  mutate(years=str_extract(title, "\\([0-9]{4}\\)"))%>%
  group_by(years)%>%
  summarize(b_y = mean(rating- mu - b_i - b_u - b_g))
```

There are several ways to evaluate a model. In this project, the root-mean-square error (RMSE) is used. For evaluating the model which is trained using only the training set, we use the test set "validation". In order to calculate the RMSE, the parameters (b_i, b_u, b_g, b_y) are joined into the test set "validation", so that the ratings can be predicted for the test set and can be compared to the true ratings of the test set.

```
#join b_i, b_u, b_g into validation set
predicted_ratings<- validation%>%
  left_join(b_i, by = "movieId")%>%
  left_join(b_u, by = "userId")%>%
  left_join(b_g, by= "genres")%>%
  mutate(years=str_extract(title, "\\([0-9]{4}\\)"))%>%
  left_join(b_y, by= "years")%>%
  mutate(pred= mu + b_i + b_u + b_g + b_y)%>%
  .$pred

#calculate RMSE
iugy_rmse<-RMSE(predicted_ratings, validation$rating)
paste0("RMSE of Movie+User+Genre+Year Model: ", iugy_rmse)
```

```
## [1] "RMSE of Movie+User+Genre+Year Model: 0.864760636190245"
```

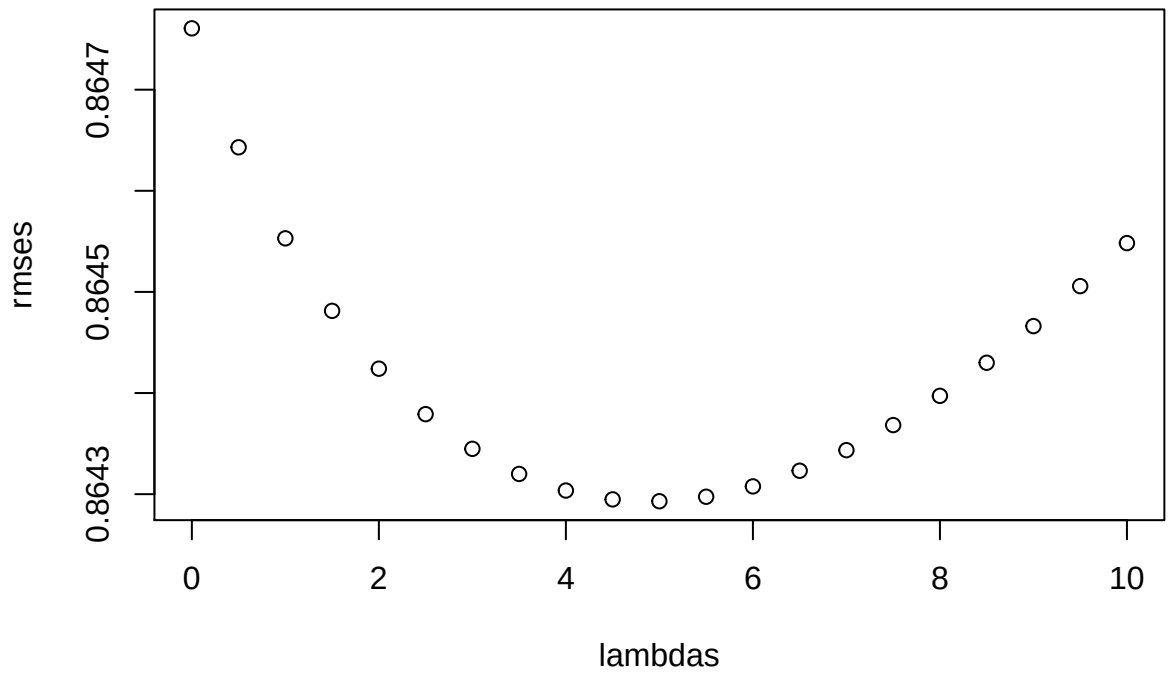
2.4 Regularization

Since there are a lot of movies that only have one or few ratings and their ratings have a high variability, they are more likely to result in a higher error when the algorithm is tested on the test set. With regularization, a penalty term is added to the objective function. The penalty term contains a variable lambda which is a tuning parameter. Thus, for the best result, the best value for the parameter has be calculated. In order to do so, the algorithm is applied on different values of lambda.

```
##Use regularization
#calculate best lambda

lambdas<- seq(0,10,0.5)

#build linear model depending on different lambdas
rmses<- sapply(lambdas, function(lambda){
```

```
  #Movie Effect
  b_i_reg<- edx%>%
    group_by(movieId)%>%
    summarize(b_i_reg = sum(rating-mu)/(n()+lambda))

  #User Effect
  b_u_reg<- edx%>%
    left_join(b_i_reg, by = "movieId")%>%
    group_by(userId)%>%
    summarize(b_u_reg = sum(rating-mu-b_i_reg)/(n()+lambda))

  #Genre Effect
  b_g_reg<- edx%>%
    left_join(b_i_reg, by = "movieId")%>%
    left_join(b_u_reg, by = "userId")%>%
    group_by(genres)%>%
    summarize(b_g_reg = sum(rating-mu-b_i_reg-b_u_reg)/(n()+lambda))

  #Year Effect (year in which the movie was released)
  b_y_reg<- edx%>%
    left_join(b_i_reg, by = "movieId")%>%
    left_join(b_u_reg, by = "userId")%>%
    left_join(b_g_reg, by= "genres")%>%
    mutate(years=str_extract(title, "\\([0-9]{4}\\)"))%>%
    group_by(years)%>%
    summarize(b_y_reg = sum(rating-mu-b_i_reg-b_u_reg-b_g_reg)/(n()+lambda))

  #join b_i, b_u, b_g into the validation set
  predicted_ratings_reg<- validation%>%
    left_join(b_i_reg, by = "movieId")%>%
    left_join(b_u_reg, by = "userId")%>%
    left_join(b_g_reg, by= "genres")%>%
    mutate(years=str_extract(title, "\\([0-9]{4}\\)"))%>%
    left_join(b_y_reg, by= "years")%>%
    mutate(pred= mu + b_i_reg + b_u_reg + b_g_reg + b_y_reg)%>%
    .$pred

  #calculate RMSE
  return(RMSE(predicted_ratings_reg, validation$rating))
})
```

In the following plot the RMSEs are plotted against the different values for lambda in order to find the best lambda.




```
## [1] "Lambda with best RMSE: 5"
```

```
## [1] "RMSE of Final Model: 0.864292990168556"
```

3 Results

The following table shows the different RMSEs resulting when using the different models. Building a linear model with one or two variables (in this case Movie and User Effect) results in a substantial improvement of the RMSE. As expected, adding more variables (here year and genre) to the model only has a small effect on the RMSE and adding regularization can improve the RMSE further by a little bit. The RMSE of the final model is: 0.8642930.

Model	RMSE
Average Rating	1.0612017
Movie Effect	0.9439087
Movie+User Effect	0.8653488
Movie+User+Genre Effect	0.8649469
Movie+User+Genre+Year Effect	0.8647606
Regularization+Movie+User+Genre+Year	0.8642930

4 Conclusion

This project uses only linear regression and regularization. However, there are more machine learning methods that could be used to improve the algorithm.