MovieLens Project Submission

Kirtimay Pendse

6/9/2020

Introduction

This project is part of the HarvardX: PH125.9x Data Science: Capstone course[1], the final course for the Data Science Professional Certificate. This project is centered around creating a movie recommendation algorithm using a subset of the MovieLens dataset.

The algorithm was developed using a training set (referred to as the 'edx' set) and was tested on the 'validation' set; then, the random mean squared error (RMSE) was calculated to evaluate the proximity of the predictions generated by the algorithm to the true values in the validation set. This report is split into four sections: first, the objective and motivation behind the project is highlighted, then exploratory data analysis is conducted, following which the modeling approach to develop the predicted movie rating algorithm is presented. Finally, the modeling resutls are presented along with a discussion on the algorithm's performance and its limitations.

Objective

The MovieLens dataset[2] is compiled by the GroupLens research group at the University of Minnesota. The dataset contains over 20 million ratings for more than 27,000 movies with around 140,000 users. The overall aim of this project is to develop an algorithm that can predict user ratings (ranging from 0.5 to 5) in the validation set using a subset of the MovieLens dataset provided by the edX staff (the 'edx' dataset).

The algorithm's performance is evaluated using the RMSE, a measure of the difference between the model's predicted values and the observed values which can be written as the following function[3], where u represents a user, i represent a movie and N is the total number of user/movie combinations:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

An 'accurate' model entails obtaining an RMSE of < 0.86490, and RMSE is the value used to calculcate the accuracy/effectiveness of the different models created.

Methods and Analysis

##Preparing the data

First, the dataset is downloaded from the MovieLens website and split into a training set (the 'edx' set) and a 'validation' set. The 'edx' dataset is further split into a train set and a test set, and is used to create the prediction algorithm. Once the desired RMSE is reached, the model is tested on the 'validation' dataset for the final analysis.

```
## Warning: package 'lubridate' was built under R version 3.6.2
```

[1] https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+1T2020/course/
[2] https://grouplens.org/datasets/movielens/latest/
[3] Taken from Prof. Irizarry's "Introduction to Data Science" 33.7.3

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: ggthemes

## Loading required package: scales

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: tidyverse

## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.3

## v tibble  3.0.1     v purrr   0.3.4
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## -- Conflicts ------------------------------------------------------------------- tidyverse_conflicts
## x lubridate::as.difftime() masks base::as.difftime()
## x readr::col_factor()      masks scales::col_factor()
## x lubridate::date()        masks base::date()
## x purrr::discard()         masks scales::discard()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()

## Loading required package: caret

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.6.2

##
## Attaching package: 'caret'

## The following object is masked _by_ '.GlobalEnv':
##
##     RMSE
```

```
## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following object is masked from 'package:purrr':
##
##     transpose

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

The 'edx' dataset was split into a train set (90% of the data in edx) and a test set (10% of the data in edx) using the following code:

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

##Exploratory Analysis

For the initial data exploration, the head() function was used to get an overall 'feel' for the data.

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

From the data frame above, it's observed that the dataset contains 6 variables, with each user and movie having a unique user ID and movie ID respectively, along with a rating assigned to each user-movie pair, and a timestamp. It's also evident that the title of each movie has the year in brackets, and multiple genres can be assigned to a movie.

Then, the class for each variable was determined.

```
##      userId     movieId      rating   timestamp       title      genres
##   "integer"   "numeric"   "numeric"   "integer" "character" "character"
```

It's observed that 'userId' and 'timestamp' are integers, 'movieId' and 'rating' are numeric, and 'title' and 'genres' are characters.

A summary of the edx dataset was also determined:

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

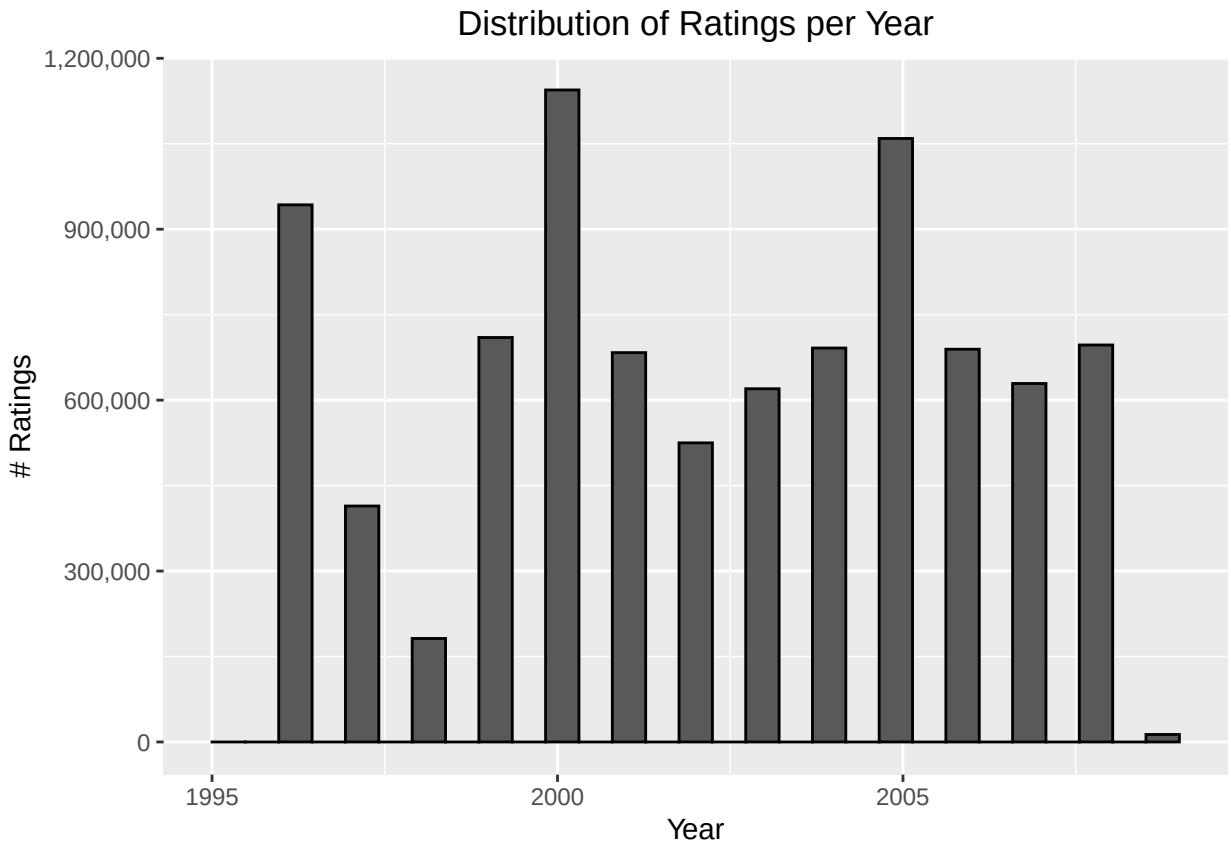
The initial observations made from the table above are that the mean rating is around 3.5, indicating a possibility that most users tend to rate movies slightly higher, and that there doesn't seem to be any missing values.

Univariate Plots

The following section inspects the variables in more detail through some plots.

Date

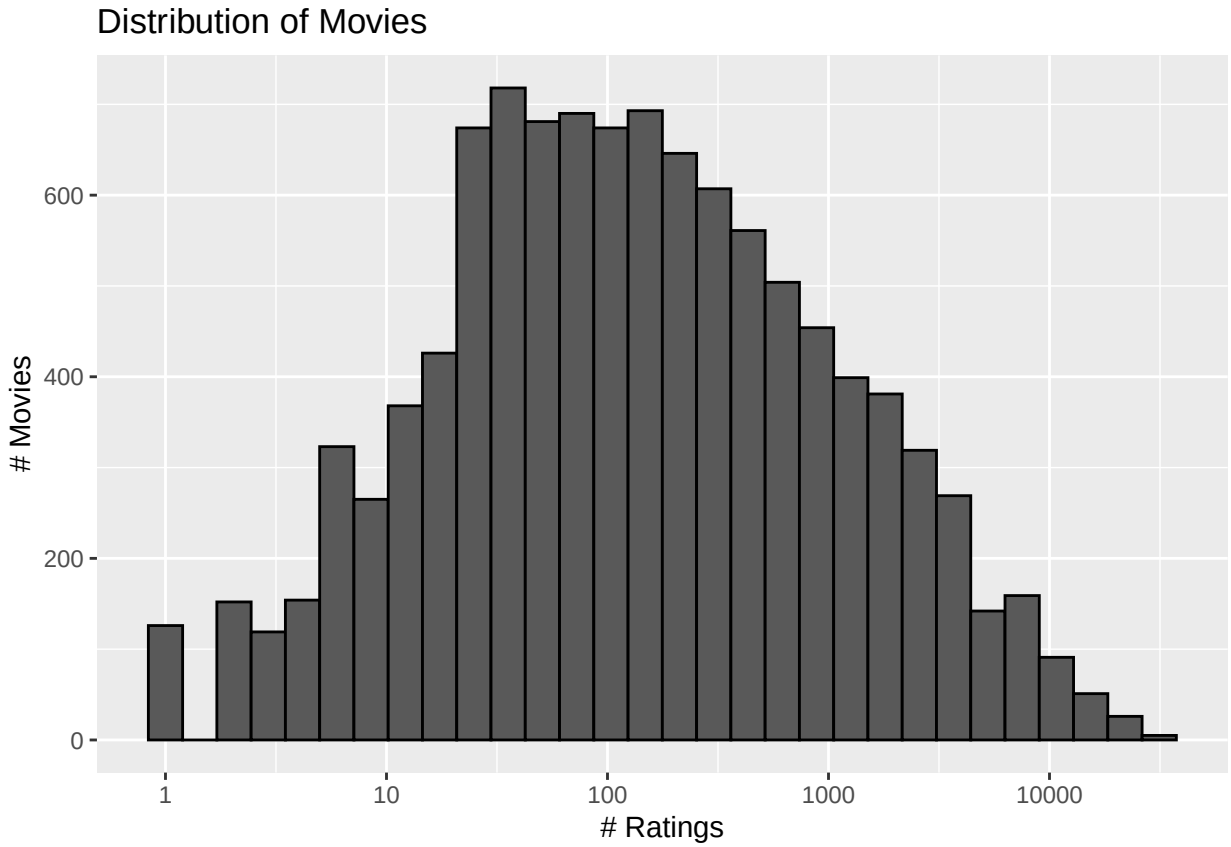
From the plot below, it's evident that the movies with the most ratings came out in 2000 and then 2005. As expected, the movies released most recently tend to have fewer ratings, but there are a few years (such as 1997 and 2002) where movies have fewer ratings.



Movies

We can observe that some movies have been rated moch often that other, while some have very few ratings and sometimes only one rating. This will be important for our model as very low rating numbers might results in untrustworthy estimate for our predictions. In fact 125 movies have been rated only once.

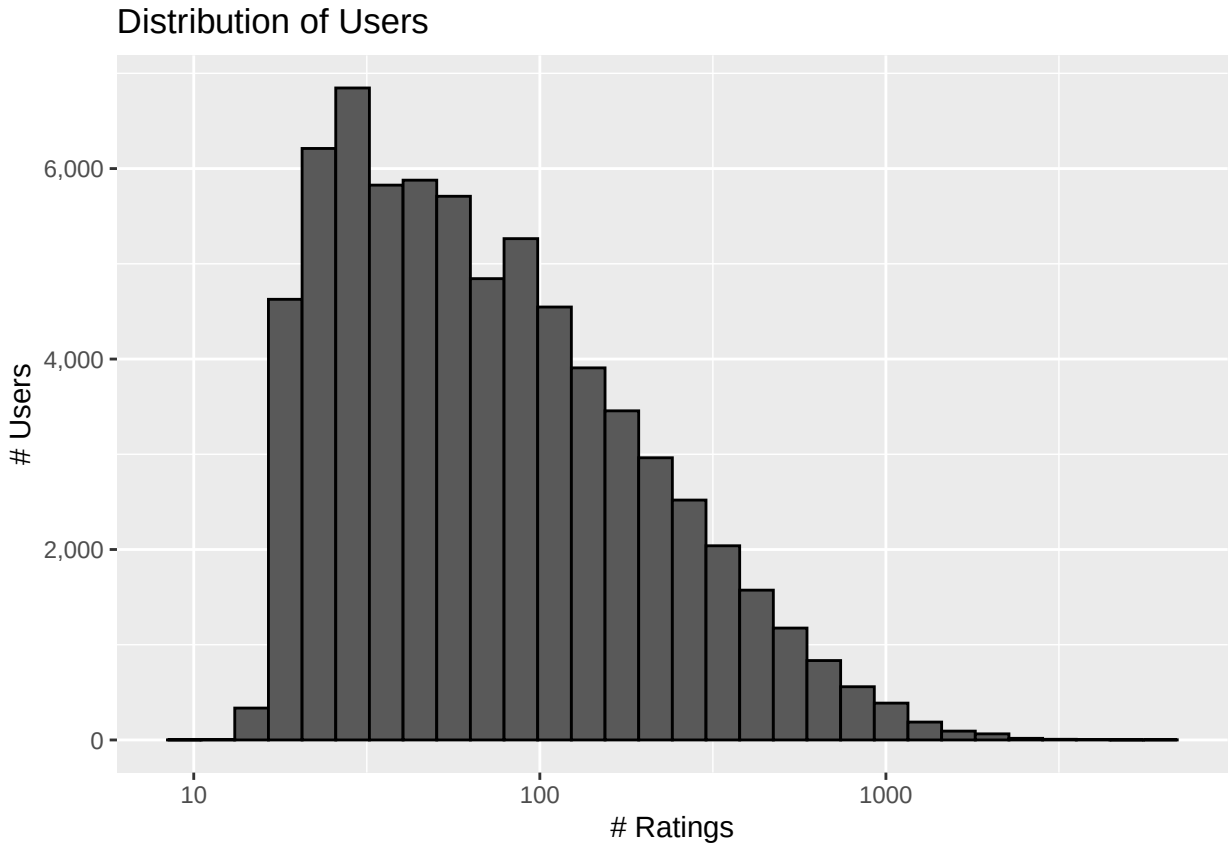
Thus regularisation and a penalty term will be applied to the models in this project. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting (the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably). Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values.




```
## # A tibble: 7 x 3
## # Groups:   date [4]
##   date       title                                                     count
##   <date>     <chr>                                                     <int>
## 1 1998-05-22 Chasing Amy (1997)                                          322
## 2 2000-11-20 American Beauty (1999)                                      277
## 3 1999-12-11 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)   254
## 4 1999-12-11 Star Wars: Episode V - The Empire Strikes Back (1980)       251
## 5 1999-12-11 Star Wars: Episode VI - Return of the Jedi (1983)           241
## 6 2005-03-22 Lord of the Rings: The Two Towers, The (2002)               239
## 7 2005-03-22 Lord of the Rings: The Fellowship of the Ring, The (2001)   227
```

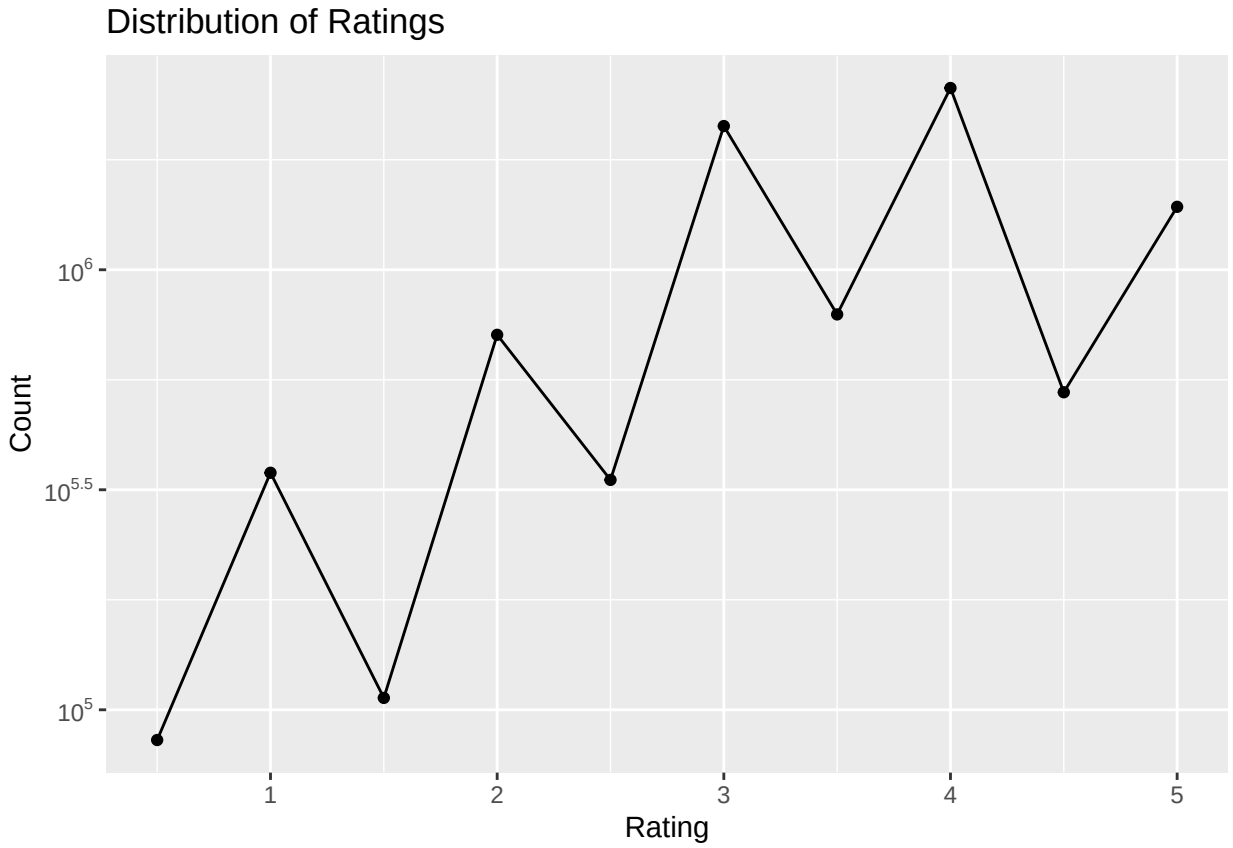
Users

We can observe that the majority of users have rated between 30 and 100 movies. So, a user penalty term need to be included later in our models.



Ratings

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. In general, half rating are less common than whole star ratings.



##Building Models

Creating a recommendation system involves the identification of the most important features that helps to predict the rating any given user will give to any movie. We start building a very simple model, which is just the mean of the observed values. Then, the user and movie effects are included in the linear model, improving the RMSE. Finally, the user and movie effects receive regularization parameter that penalizes samples with few ratings.