HarvardX: PH125.9x Data Science
MovieLens Rating Prediction Project

Subrahmanyam Aryasomayajula

March 13, 2021

Introduction

Movie lens prediction project is done as a part of Data Science , HarvardX: PH125.9x. Various machine learning models are analyzed , developed and explored to build recommendation systems. Movie Recommendation systems use ratings that users have given movies to make specific recommendations to the users regarding the movies they would like to watch.Machine learning models are based on algorithms built with data . For this project we are provided with MovieLens 10M dataset by grouplens resarch at http://files.grouplens.org/datasets/movielens/ml-10m.zip . This course also refers to and draws in on the knowledge developed as a part of "Data Science professional certificate program" by Harvardx.

Overview

This project originally came out a challenge Netflix offered to the data science community to improve their recommendation algorithm by 10% and win a million dollars.We have explored Various machine learning models to develop the recommendation algorithm based on the ratings users
give to the movies .We divide the data into training set and validation set.We train our machine learning models using the training data set and use validation data set to make predictions.Residual mean squared error (RMSE) on a validation set is evaluated to determine the efficiency of the model . The RMSE is defined as:

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

We compute the RMSE error for each of the models developed on the validation set . The objective is to minimize the RMSE error . The Machine learning model with the minimum RMSE errors does a better prediction than the other models and would be a better recommendation system.

Executive Summmary

The goal of the project is to develop a better movie recommendation system .This is accomplished by analysis and development of various machine learning models and applying them to movies dataset to make prediction . The accuracy of our prediction is measured by the Residual mean squared error (RMSE). RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, a lower RMSE is better than a higher one. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers. Four ML models are developed and compared using their resulting RMSE in order to assess their quality. The goal of algorithm is to obtain a RMSE to be lower than 0.857. The function that

computes the RMSE for vectors of ratings and their corresponding predictors is

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

The model which results in a lowest RMSE value is a better model of the four models analyzed .

Dataset preparation

The dataset is downloaded from the movie lens research website . A sample set of 10 Million records are downloaded . The data downloaded is split into two data sets . edx dataset which is the training dataset and validation dataset. The algorithm is trained with the edx dataset and tested with the validation dataset. Validation dataset is 10% of the movie lens dataset.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()

download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)



ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))



movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)



colnames(movies) <- c("movieId", "title", "genres")



#if using R 4.0 or later:
```

```
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))




movielens <- left_join(ratings, movies, by = "movieId")

head(movielens)
```

```
##    userId movieId rating timestamp                          title
## 1:      1     122      5 838985046                Boomerang (1992)
## 2:      1     185      5 838983525                 Net, The (1995)
## 3:      1     231      5 838983392            Dumb & Dumber (1994)
## 4:      1     292      5 838983421                 Outbreak (1995)
## 5:      1     316      5 838983392                 Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1:               Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                       Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)

edx <- movielens[-test_index,]

temp <- movielens[test_index,]



# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
#rm(dl, ratings, movies, test_index, temp, movielens, removed)



edx %>% summarize(n_users = n_distinct(userId),n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

Dataset exploration

We can see from the above section that there are 69878 distinct users and 10677 distinct movies. We can also see that all the fields in the movie lens dataset has values populated .The field names in the movie lens dataset are userId, movieId, rating, timestamp ,title , genres.

Loss function

Loss function used to determine the error loss is RMSE .We determine the best algorithm based on the residual mean squared error (RMSE) on a validation set. Here in 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:

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

with N being the number of user/movie combinations and the sum occurring over all these combinations.

RMSE is similar to standard deviation: it is the error we make when predicting a movie rating. Let's write a function that computes the RMSE for vectors of ratings and their corresponding predictors:

Machine Learning Models explored.

In this project we have analyzed four machine learning models and calculated the RMSE error for each of these models . The four machine learning models are : 1. Simple machine learning model 2. Movie effects model 3. User effects model
4. Penalized least squares estimates model .

Analysis of each of the models is presented in below sections.

Simple recommendation model

A model that assumes the same rating for all movies and users with all the differences explained by random variation would look like this:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

with $\varepsilon_{i,u}$ independent errors sampled from the same distribution centered at 0 and μ the "true" rating for all movies. The estimate that minimizes the RMSE is the least squares estimate of μ , for this case is the average of all ratings:

```
#############################################################################
## Simple recommendation model
#############################################################################

mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

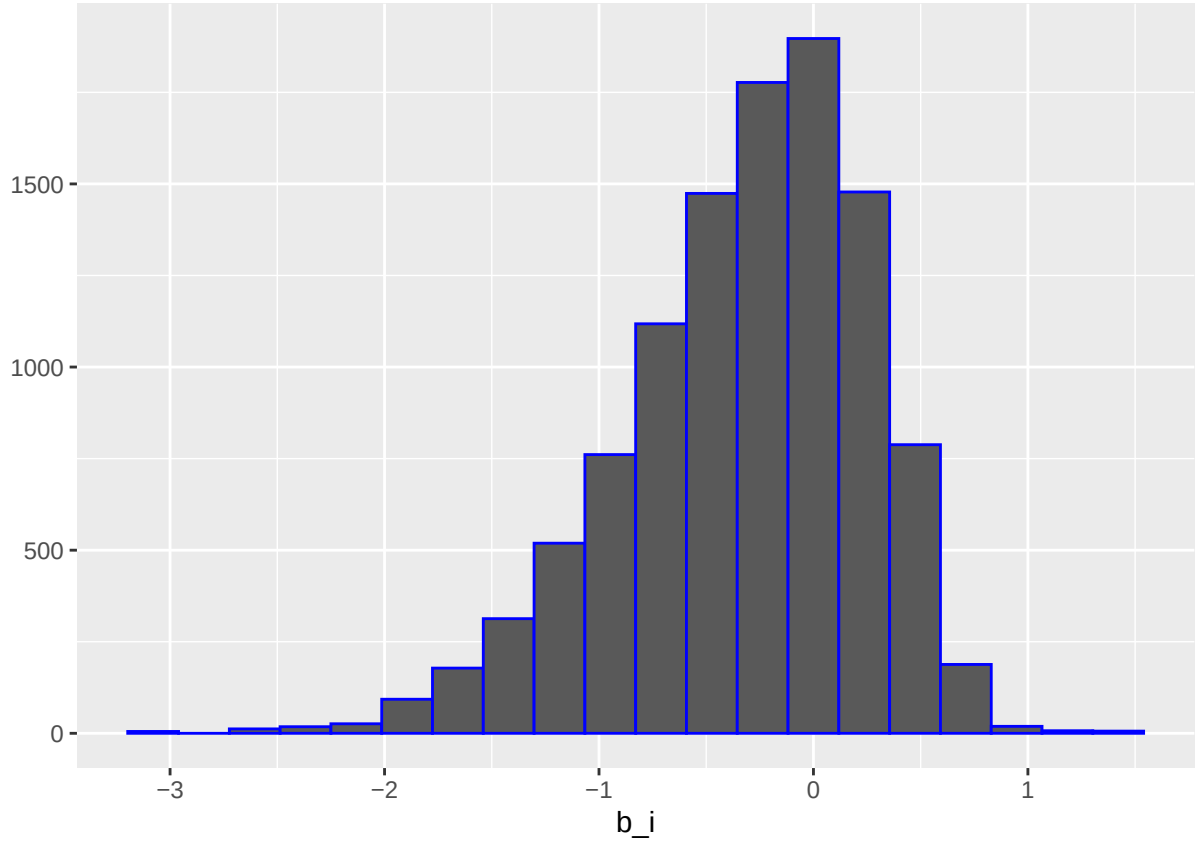
```
naive_rmse <- RMSE(validation$rating, mu)
cat("naive_rmse: ", naive_rmse)
```

```
## naive_rmse:  1.061202
```

Our goal is to get an RMSE of about 0.857 or better.

Modeling movie effects

```
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```



We know that different movies are rated differently from experience. Some movies are rated higher than others and some lower etc . Data actually confirms this . Simple recommendation model is amended by adding the term $b_i$to represent average ranking for movie i:

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

bs s referred as effects or sometimes as "bias", thus the b notation.

We can again use least squares to estimate the b_i in the following way:

fit <- lm(rating ~ as.factor(movieId), data = movielens)

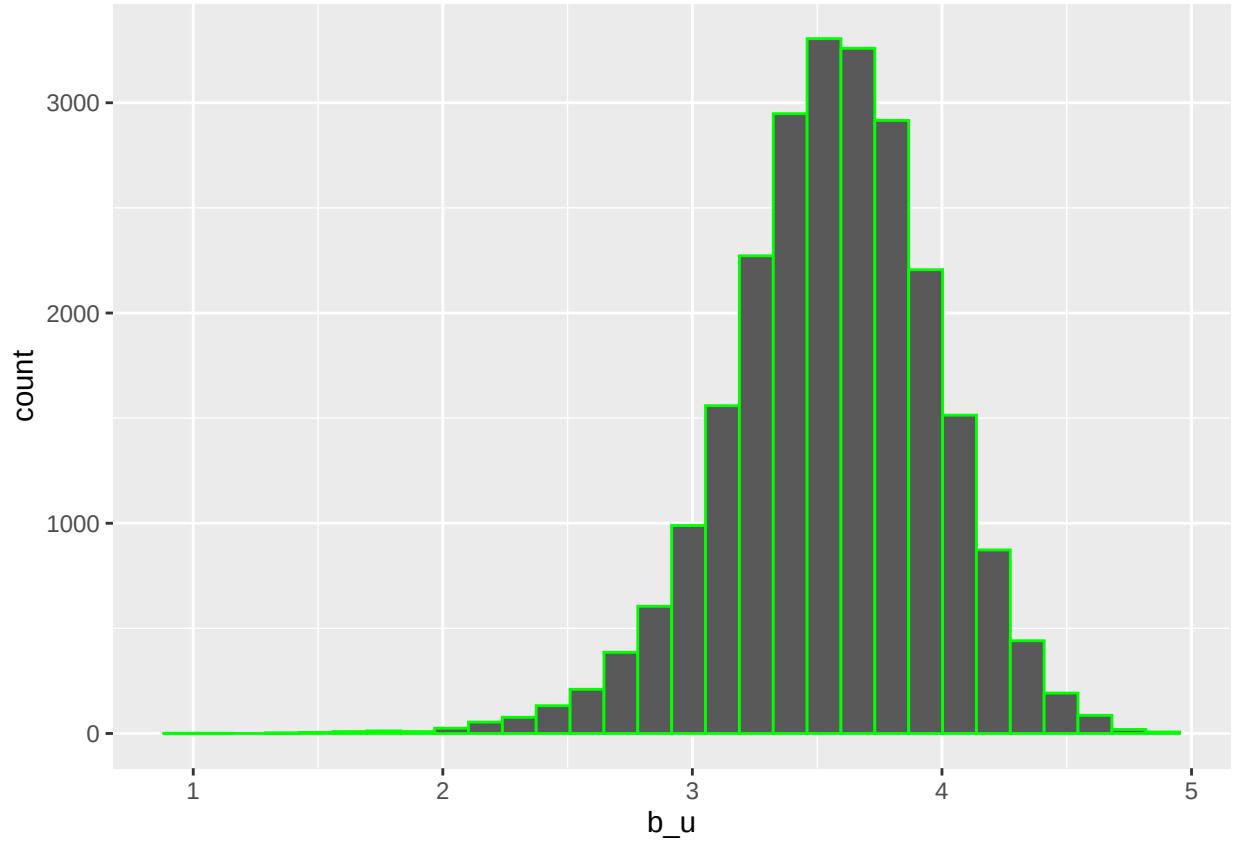
'lm() function is slow to run . For this model we know that the least squares estimate $\hat{b}_i$ is just the average of $Y_{u,i} - \hat{\mu}$ for each movie i. we use this model to compute our RMSE .

b_i = 1.5 s a perfect 5.0 rating as mean mu = 3.5.

```
##########################################################################
### Method 2
## effects or bi . Bias ,b notation  movie effects
##########################################################################

predicted_ratings <- mu + validation  %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

movie_effects_rmse <- RMSE(predicted_ratings, validation$rating)

cat("movie_effects_rmse: ", movie_effects_rmse)
```

```
## movie_effects_rmse:  0.9439087
```

Modeling User effects.



Here we computed the average rating for user u for those that have rated 100 or more movies:

Here some users with different rating standards may be rating movies differently than an average user. Some may not like most movies , some may like every movie . We account for such effects in this model.

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

where b_u is a user-specific effect.

To fit this model, we could again use `lm` like this:

lm(rating ~ as.factor(movieId) + as.factor(userId))

but as it is slowto run the model, we compute an approximation by computing $\hat{\mu}$ and $\hat{b}_i$ and estimating $\hat{b}_u$ as the average of $y_{u,i} - \hat{\mu} - \hat{b}_i$:

```
#####################################################################
### Method 3 : User  effects

#####################################################################

user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))



predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

user_specific_rmse <- RMSE(predicted_ratings, validation$rating)

cat("user_specific_rmse: ", user_specific_rmse)
```

```
## user_specific_rmse:  0.8653488
```

Exploration

Below we have listed ten largest mistakes with residual error . Ten best movies according to our estimate.Ten worst movies according to our estimate. We have also noticed that the supposed “best” and “worst” movies were rated by very few users. These movies were mostly obscure ones. This is because with just a few users, we have more uncertainty. These are noisy estimates which result in erroneous predictions .

```
#################################################################
###  Regularization
#################################################################


validation %>%
  left_join(movie_avgs, by='movieId') %>%
  mutate(residual = rating - (mu + b_i)) %>%
  arrange(desc(abs(residual))) %>%
```

```
  slice(1:10) %>%
  pull(title)
```

```
##  [1] "PokÃ©mon Heroes (2003)"           "Shawshank Redemption, The (1994)"
##  [3] "Shawshank Redemption, The (1994)" "Shawshank Redemption, The (1994)"
##  [5] "Godfather, The (1972)"           "Godfather, The (1972)"
##  [7] "Godfather, The (1972)"           "Usual Suspects, The (1995)"
##  [9] "Usual Suspects, The (1995)"      "Usual Suspects, The (1995)"
```

```
movie_titles <- movies %>%
  select(movieId, title) %>%
  distinct()


movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10)  %>%
  pull(title)
```

```
##  [1] "Hellhounds on My Trail (1999)"
##  [2] "Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)"
##  [3] "Shadows of Forgotten Ancestors (1964)"
##  [4] "Fighting Elegy (Kenka erejii) (1966)"
##  [5] "Sun Alley (Sonnenallee) (1999)"
##  [6] "Blue Light, The (Das Blaue Licht) (1932)"
##  [7] "Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)"
##  [8] "Human Condition II, The (Ningen no joken II) (1959)"
##  [9] "Human Condition III, The (Ningen no joken III) (1961)"
## [10] "Constantine's Sword (2007)"
```

```
movie_avgs %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  slice(1:10)  %>%
  pull(title)
```

```
##  [1] "Besotted (2001)"
##  [2] "Hi-Line, The (1999)"
##  [3] "Accused (Anklaget) (2005)"
##  [4] "Confessions of a Superhero (2007)"
##  [5] "War of the Worlds 2: The Next Wave (2008)"
##  [6] "SuperBabies: Baby Geniuses 2 (2004)"
##  [7] "Hip Hop Witch, Da (2000)"
##  [8] "Disaster Movie (2008)"
##  [9] "From Justin to Kelly (2003)"
## [10] "Criminals (1996)"
```

```
edx %>% count(movieId) %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  slice(1:10) %>%
  pull
```

```
##  [1] "Hellhounds on My Trail (1999)"
##  [2] "Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)"
##  [3] "Shadows of Forgotten Ancestors (1964)"
##  [4] "Fighting Elegy (Kenka erejii) (1966)"
##  [5] "Sun Alley (Sonnenallee) (1999)"
##  [6] "Blue Light, The (Das Blaue Licht) (1932)"
##  [7] "Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)"
##  [8] "Human Condition II, The (Ningen no joken II) (1959)"
##  [9] "Human Condition III, The (Ningen no joken III) (1961)"
## [10] "Constantine's Sword (2007)"
```

```
edx %>% count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  slice(1:10) %>%
  pull(n)
```

```
## Joining, by = "movieId"
```

```
##  [1]   2   1   1   1   2  56  14  32 199   2
```

Penalized least squares

Idea behind regularization is to constrain the total variability of the movie effect sizes. $\sum_{i=1}^{5} b_i^2$. Specifically, instead of minimizing the least squares equation, we minimize an equation that adds a penalty:

$$\sum_{u,i} (y_{u,i} - \mu - b_i)^2 + \lambda \sum_i b_i^2$$

The first term is just the sum of squares and the second is a penalty that gets larger when many b_i are large. Using calculus we can actually show that the values of b_i that minimize this equation are:

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

where n_i is the number of ratings made for movie i. This approach will have our desired effect: when our sample size n_i is very large,a case which will give us a stable estimate, then the penalty λ is effectively ignored since $n_i + \lambda \approx n_i$. However, when the n_i is small, then the estimate $\hat{b}_i(\lambda)$ is shrunken towards 0. The larger λ, the more we shrink.

```
#########################################################################
## Method 4
#Let's compute these regularized estimates of  bi  using  = 3 . Later, we will see why we picked 3.

##Regularization,Penalized  least  squares estimates with full  cross validaton.

#########################################################################

lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
```

```
  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
})


penalized_least_squares_lambda <- lambdas[which.min(rmses)]
penalized_least_squares_lambda
```

```
## [1] 5.25
```

```
penalized_least_squares_lambda_rmse = min(rmses)

cat("penalized_least_squares_lambda_rmse: ", penalized_least_squares_lambda_rmse)
```

```
## penalized_least_squares_lambda_rmse:  0.864817
```

Results

RMSE values from all the four machine learning models are printed below . The four machine learning models are , Simple machine learning model , Movie effects model , effects model and penalized least squares estimates model. We can clearly see that , penalized least squares estimates model has the lowest RMSE value.

```
Method <- c("naive_rmse", "movie_effects_rmse", "user_specific_rmse", "penalized_least_squares_lambda_r
RMSE <- c(naive_rmse,movie_effects_rmse,user_specific_rmse,penalized_least_squares_lambda_rmse)
df<- class.df<- data.frame(Method,RMSE)

df
```

```
##                               Method      RMSE
## 1                         naive_rmse 1.0612018
## 2                movie_effects_rmse 0.9439087
## 3                user_specific_rmse 0.8653488
## 4 penalized_least_squares_lambda_rmse 0.8648170
```

Conclusion

This project is part of Data Science , HarvardX: PH125.9x. Here in we have studied machine learning models by applying them to a movie recommendation system . We have analyzed four machine learning models are , Simple machine learning model , Movie effects model , effects model and penalized least squares estimates model. Residual mean squared error is used as the measure of the accuracy of the models. We have found that penalized least squares estimates model has the lowest RMSE value and is a better model of the four models considered for a movie recommendation system . We can further improve the RMSE if we can take into account additional profile characteristics of the user such as gender , age , language , nationality etc as the ratings tend to be more accurate .