MovieLens Project Report

Anthony Bouz

2023-12-05

Introduction :

In the Capstone course of the HarvardX Professional Certificate in Data Science (PH125.9x), we will explore and visually examine the MovieLens data set of GroupLens Research which features over 10 million film ratings. The objective will be to develop a machine-learning model by creating training and test sets to predict movie ratings on a validation set that achieves a Root Mean Square Error (RMSE).The regularised movie and user effect model will be used that uses regularisation to reduce overfitting and capture individual movie and user biases.

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

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.3.1
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
```

```
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

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

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.3.1
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift

library(tidyverse)
library(caret)

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

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip",
dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"),
simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"),
simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

```r
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later

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

# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)

## Joining with `by = join_by(userId, movieId, rating, timestamp, title,
genres)`

edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Exploratory Data Analysis

```r
str(edx)

## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392
838984474 838983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)"
"Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Action|Drama|Sci-Fi|Thriller" "Action|Adventure|Sci-Fi" ...
```

There are 9,000,055 observations and 6 columns. Each observation represents a rating given by one user for one movie. Columns include userId, movieId, rating, timestamp, title and genres.

```r
n_distinct(edx$movieId)

## [1] 10677
```

The dataset has 10677 unique movies

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

The dataset has 69878 unique users

```
n_distinct(edx$genres)
```

```
## [1] 797
```

The dataset has 797 unique genres

```
library(stringr)
edx$year <- str_extract(edx$title, "\\((\\d{4})\\)")
edx$year <- as.numeric(gsub("\\D", "", edx$year))
range(edx$year)
```

```
## [1] 1915 2008
```

So the movies are based from years 1915 to 2008

```
genre_count = edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(Count = n()) %>%
  arrange(desc(Count))

unique(genre_count$genres)
```

```
##  [1] "Drama"              "Comedy"             "Action"
##  [4] "Thriller"           "Adventure"          "Romance"
##  [7] "Sci-Fi"             "Crime"              "Fantasy"
## [10] "Children"           "Horror"             "Mystery"
## [13] "War"                "Animation"          "Musical"
## [16] "Western"            "Film-Noir"          "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

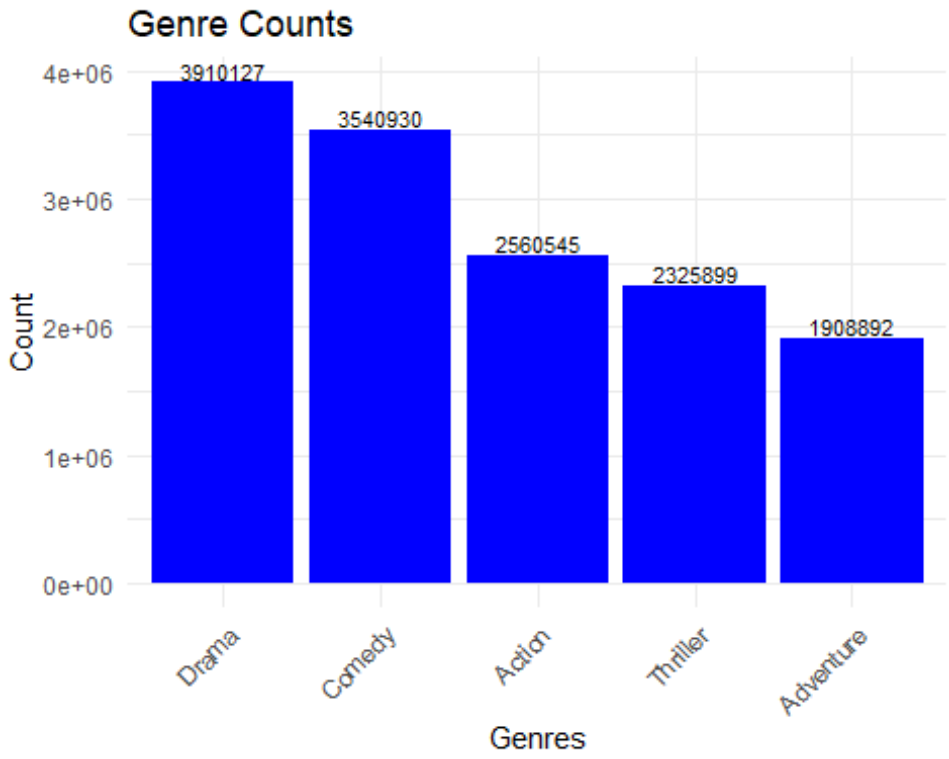
```
genre_count %>% slice_head(n = 5)
```

```
## # A tibble: 5 × 2
##   genres      Count
##   <chr>       <int>
## 1 Drama     3910127
## 2 Comedy    3540930
## 3 Action    2560545
## 4 Thriller  2325899
## 5 Adventure 1908892
```

There are total 18 unique Genres leaving out IMAX and (no genres listed)

“Drama”,“Comedy”,“Action”,“Thriller”,“Adventure” are the top 5 go to genres for users with Drama being the most preferred

```r
genre_count = data.frame(Genres =
c("Drama","Comedy","Action","Thriller","Adventure"),
                         Count = c(3910127,3540930,2560545,2325899,1908892))

genre_count <- genre_count[order(-genre_count$Count), ]

ggplot(genre_count, aes(x = reorder(Genres, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Genre Counts",
       x = "Genres",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = Count), vjust = -0.2, size = 3)
```



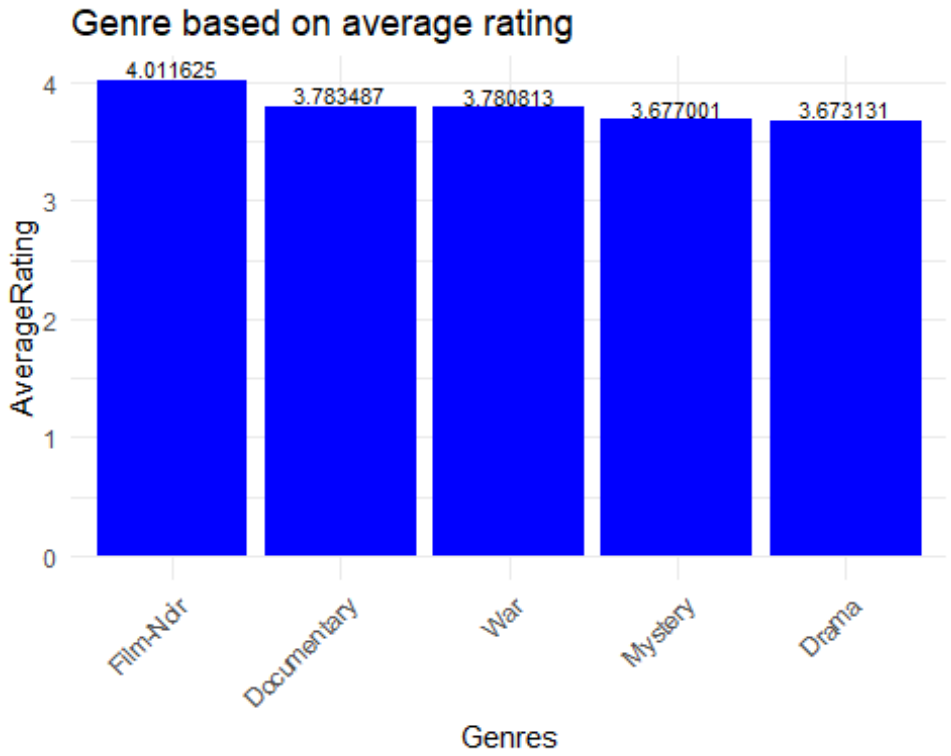

```r
average_rating_per_genre <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(AverageRating = mean(rating, na.rm = TRUE))

average_rating_per_genre %>% arrange(desc(AverageRating)) %>%  filter(genres
!= "IMAX" & genres != "(no genres listed)") %>% slice_head(n = 5)
```

```
## # A tibble: 5 × 2
##   genres      AverageRating
##   <chr>               <dbl>
```

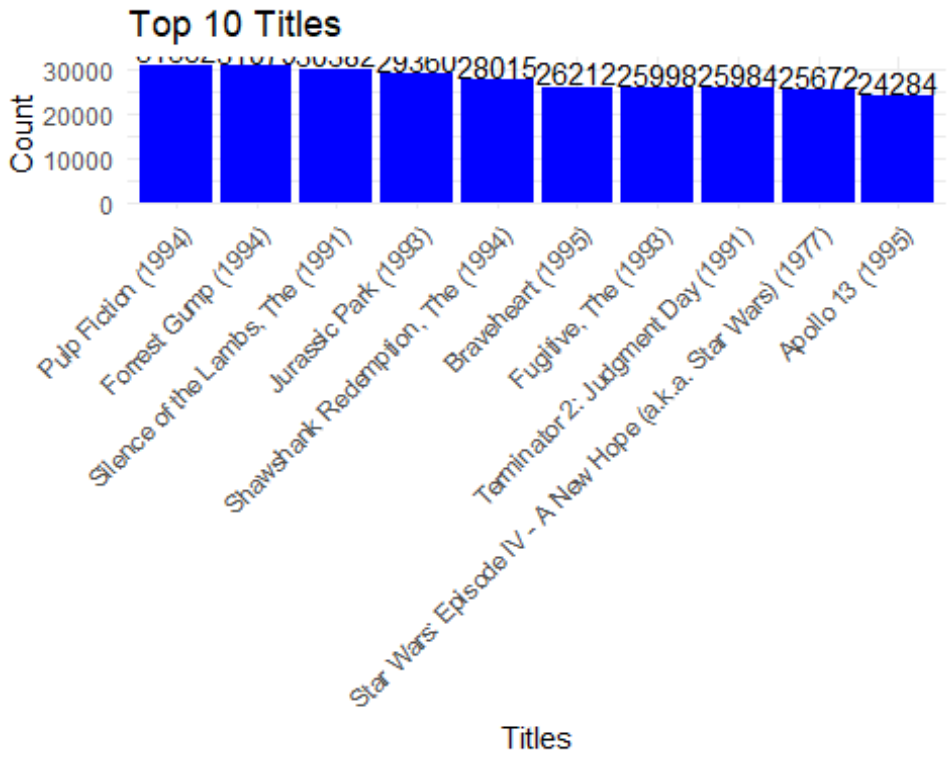
```
## 1 Film-Noir            4.01
## 2 Documentary          3.78
## 3 War                  3.78
## 4 Mystery              3.68
## 5 Drama                3.67
```

These are the top 5 genres with highest average rating

```
genre_rating_count = data.frame(Genres = c("Film-
Noir","Documentary","War","Mystery","Drama"),
                                AverageRating =
c(4.011625,3.783487,3.780813,3.677001,3.673131))

genre_rating_count <- genre_rating_count[order(-
genre_rating_count$AverageRating), ]

ggplot(genre_rating_count, aes(x = reorder(Genres, -AverageRating), y =
AverageRating)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Genre based on average rating",
       x = "Genres",
       y = "AverageRating") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = AverageRating), vjust = -0.2, size = 3)
```

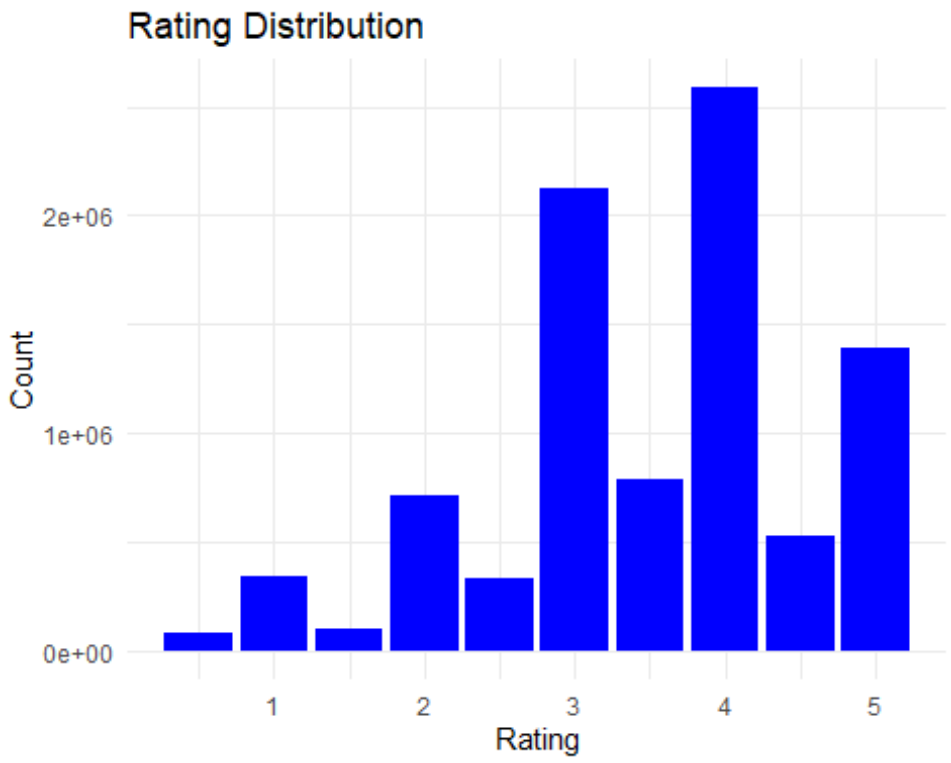


```
top_10_titles <- edx %>%
  group_by(title) %>%
  summarize(Count = n()) %>%
  arrange(desc(Count)) %>%
  head(10)

ggplot(top_10_titles, aes(x = reorder(title, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Top 10 Titles",
       x = "Titles",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = Count), vjust = -0.1, size = 4,font="bold")

## Warning in geom_text(aes(label = Count), vjust = -0.1, size = 4, font =
## "bold"): Ignoring unknown parameters: `font`
```



These are the top 10 popular movies

```
ggplot(edx, aes(x = rating)) +
  geom_bar(stat = "count", fill = "blue") +
  labs(title = "Rating Distribution",
       x = "Rating",
       y = "Count") +
  theme_minimal()
```



People prefer to rate movies with a max rating of 4

Data Modelling

The regularised movie and user effect model uses regularisation to reduce overfitting and capture individual movie and user biases. Finding a balance between fitting the training data effectively and generalising to new, unobserved data is aided by the regularisation term.

```
set.seed(123)
train_indices <- sample(1:nrow(edx), 0.8 * nrow(edx))
train_data <- edx[train_indices, ]
test_data <- edx[-train_indices, ]
lambdas <- seq(0, 10, 1)
rmses <- sapply(lambdas, function(l) {

  mu <- mean(edx$rating)

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

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

```
  predicted_ratings <-
    test_data %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>% .$pred

  return(sqrt(mean((predicted_ratings- test_data$rating)^2,na.rm = TRUE)))
})

min(rmses)  # for test data
```

```
## [1] 0.8661309
```

```
lambdas[which.min(rmses)]
```

```
## [1] 5
```

I splitted the dataset (edx) into two parts: a training set (train_data) and a test set (test_data). This ensures that I have a separate portion of the data to evaluate the collaborative filtering model's performance.

I created a sequence of regularization parameters (lambdas) from 0 to 10 with a step size of 1. These lambdas will help me control the regularization strength in my collaborative filtering model.

I compute the Root Mean Squared Error (RMSE) between my predicted and actual ratings on the test set. 0.8661309 is the minimum RMSE on test_data , and the minimum lambda is found out to be 5

RMSE for final_holdout_test with optimal lambda = 5

```
mu <- mean(edx$rating)
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + 5))

b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() +5))

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

Rmse = sqrt(mean((predicted_ratings- final_holdout_test$rating)^2,na.rm =
TRUE))
Rmse
```

```
## [1] 0.8648177
```

I then using lambda = 5 train on the whole data predicting on final_holdout_test that gives me 0.8648177

Conclusion :

With an RMSE of 0.8648177, the final model—which combined UserId and MovieId Effects with Regularization—performed admirably, but further biases might be investigated to raise the model's accuracy even further.Future versions of the K-Nearest-Neighbors and Collaborative Filtering using cosine similarity models probably promise to keep improving and enhancing the overall experience for streamers worldwide.