MovieLens, Harvard Capstone

Hamza Khan

1) Indtroduction

The MovieLens dataset is a publicly available dataset generated by the GroupLens Research lab at the University of Minnesota. It is one of the most widely used datasets for the development and evaluation of recommender systems. The dataset contains movie ratings, movie information, and user information. The data includes several versions of the dataset with different amounts of ratings and users, ranging from 100,000 ratings by 700 users to 26 million ratings by 138,000 users. The movie information includes the title, genre, and release year.

2) Methodolgy and Results

The data set contains a zip file named "ml-10M100K" which is downloaded in the main file directory. The files are later unzipped and data is extracted. This leads to our analysis which is divided majorly into three parts: • Exploratory Data Analysis • Preprocessing • Prediction and Regularization All three parts are inter-related and inter-dependent on each other and are not completely separated from each other and they do overlap among each other (i.e. some pre-processing has been done in the Exploratory Data Analysis portion).

Following code was provided by the HarvardX to download the files and turn it into data set.

```
# Create edx and final_holdout_test sets
##########################################################

# 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")

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)
```

```r
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")

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

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")

library(ggplot2)
library(dplyr)
library(ModelMetrics)
```

3) Exploring the dataset

```r
#printing first five rows of the Edx dataset.
head(edx,  n=5)
```

```
##   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)
##                          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
```

```r
summary(edx)
```

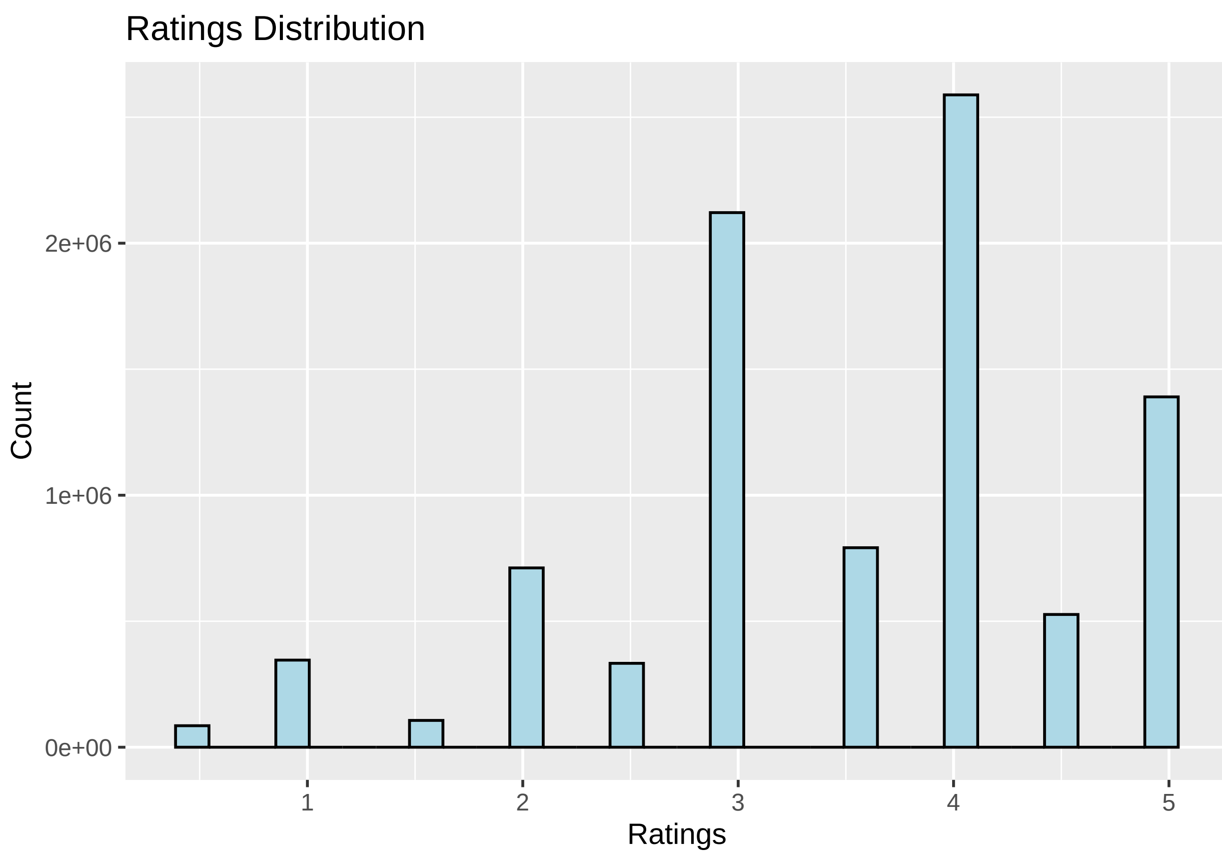
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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:9000047     Length:9000047
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```r
print(paste0('We have ', n_distinct(edx$userId), ' distinct users, ',
             n_distinct(edx$movieId), ' movies and ',
             n_distinct(edx$genres), ' genres in the dataset'))
```

Calculating distint users, movies and genres

```
## [1] "We have 69878 distinct users, 10669 movies and 797 genres in the dataset"
```

```r
# histogram of ratings

ggplot(edx, aes(x=rating)) + geom_histogram(fill="lightblue", color="black") +
  labs(x="Ratings", y="Count") +
  ggtitle("Ratings Distribution")
```




```
print(paste0("Mean Ratings: ", round(mean(edx$rating),2)))
```

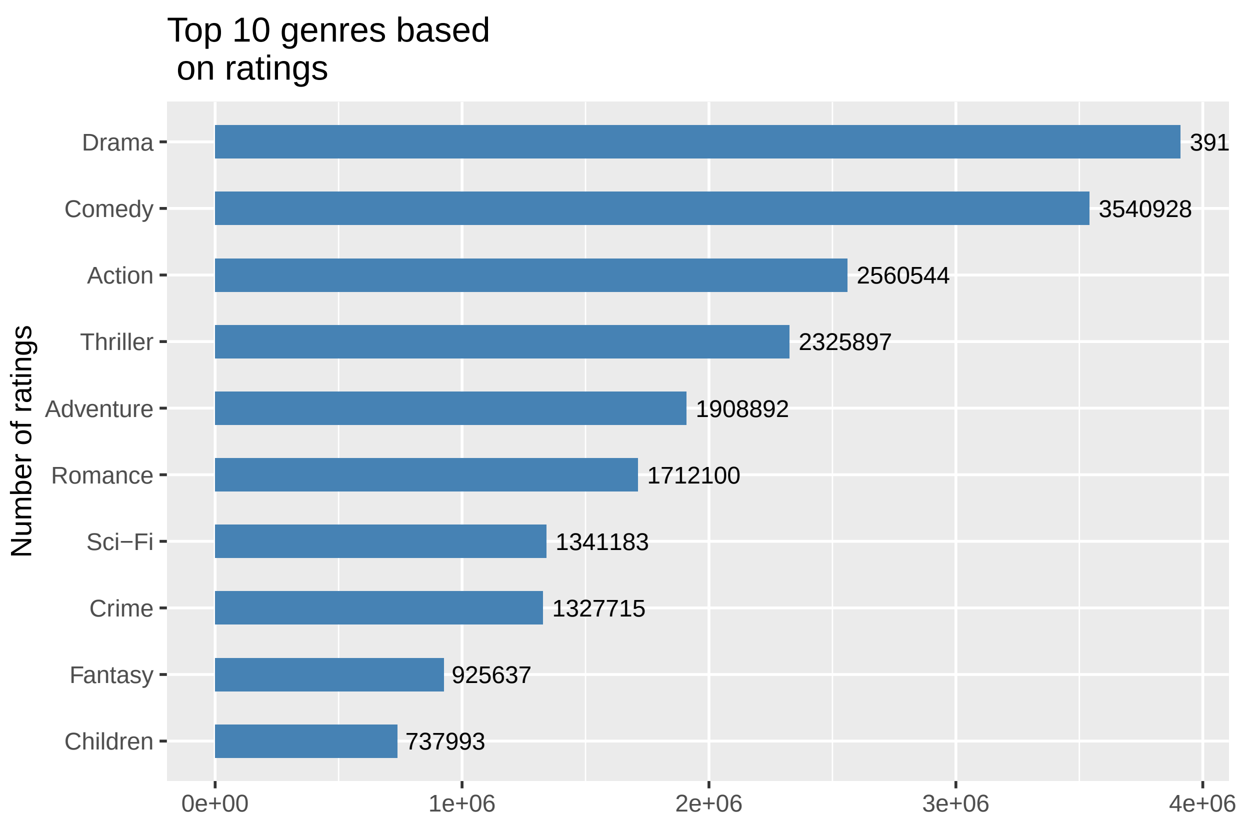
```
## [1] "Mean Ratings: 3.51"
```

```
# Separate genres into separate rows
edx_split <- edx %>%
  separate_rows(genres, sep = "\\|")

# Count the occurrences of each genre
genre_counts <- edx_split %>%
  count(genres)

# Sort by number of occurrences and display the top 10 genres
top_genres <- genre_counts %>%
  arrange(desc(n)) %>%
  top_n(10)

print(top_genres)
```

```
## # A tibble: 10 x 2
##    genres          n
##    <chr>       <int>
##  1 Drama     3910124
```

```
##   2 Comedy    3540928
##   3 Action    2560544
##   4 Thriller  2325897
##   5 Adventure 1908892
##   6 Romance   1712100
##   7 Sci-Fi    1341183
##   8 Crime     1327715
##   9 Fantasy    925637
##  10 Children   737993
```

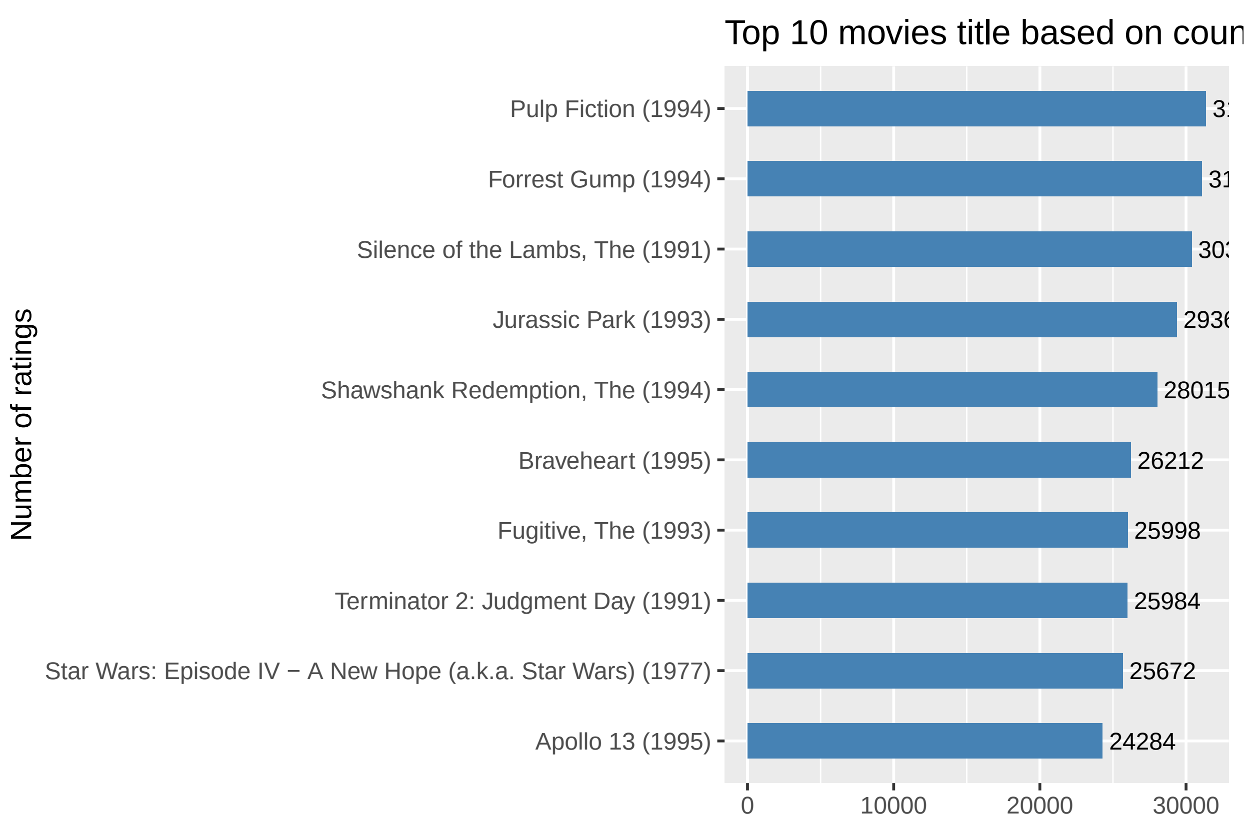
```
#plotting genres based on ratings
ggplot(top_genres, aes(x=n, y=reorder(genres,n)))+
  geom_bar(stat='identity', fill="steelblue", width = 0.5)+
  labs(x="", y="Number of ratings", title="Top 10 genres based \n on ratings") +
  geom_text(aes(label= n), hjust=-0.1, size=3)
```




```
# Count the occurrences of each movie
movie_counts <- edx %>%
  count(title)

# Sort by number of occurrences and display the top 10 movies
top_movies <- movie_counts %>%
  arrange(desc(n)) %>%
  top_n(10)

print(top_movies)
```

```
##                                                        title     n
## 1                                        Pulp Fiction (1994) 31362
## 2                                        Forrest Gump (1994) 31079
## 3                           Silence of the Lambs, The (1991) 30382
## 4                                       Jurassic Park (1993) 29360
## 5                           Shawshank Redemption, The (1994) 28015
## 6                                          Braveheart (1995) 26212
## 7                                       Fugitive, The (1993) 25998
## 8                          Terminator 2: Judgment Day (1991) 25984
## 9  Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10                                          Apollo 13 (1995) 24284
```

```r
#plotting top movies based on ratings
ggplot(top_movies, aes(x=n, y=reorder(title,n)))+
  geom_bar(stat='identity', fill="steelblue", width = 0.5)+
  labs(x="", y="Number of ratings", title="Top 10 movies title based on count based on ratings") +
  geom_text(aes(label= n), hjust=-0.1, size=3)
```



4) Data Pre-Processing

```r
#creating a dataset that will be used for modeling.
df <- edx[,colnames(edx)!="timestamp"]
df$year_released <- gsub(".*\\(([0-9]{4})\\).*", "\\1", df$title)
```

5) Modelling, Regularization and Performance

i. Modelling: Simple Regression

Equating: rating = + bi + bu + u

where, = mean ratings bi = movie effect bu = user effect

```
# calculate the average of all ratings of the df data set
mu <- mean(df$rating)

# Calculate b_i and b_u
b_i <- df %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

b_u <- df %>%
  left_join(b_i, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

# Predict ratings
predicted_ratings_bu <- df %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# Calculate RMSE
RMSE(predicted_ratings_bu, edx$rating)
```

```
## [1] 0.8567042
```

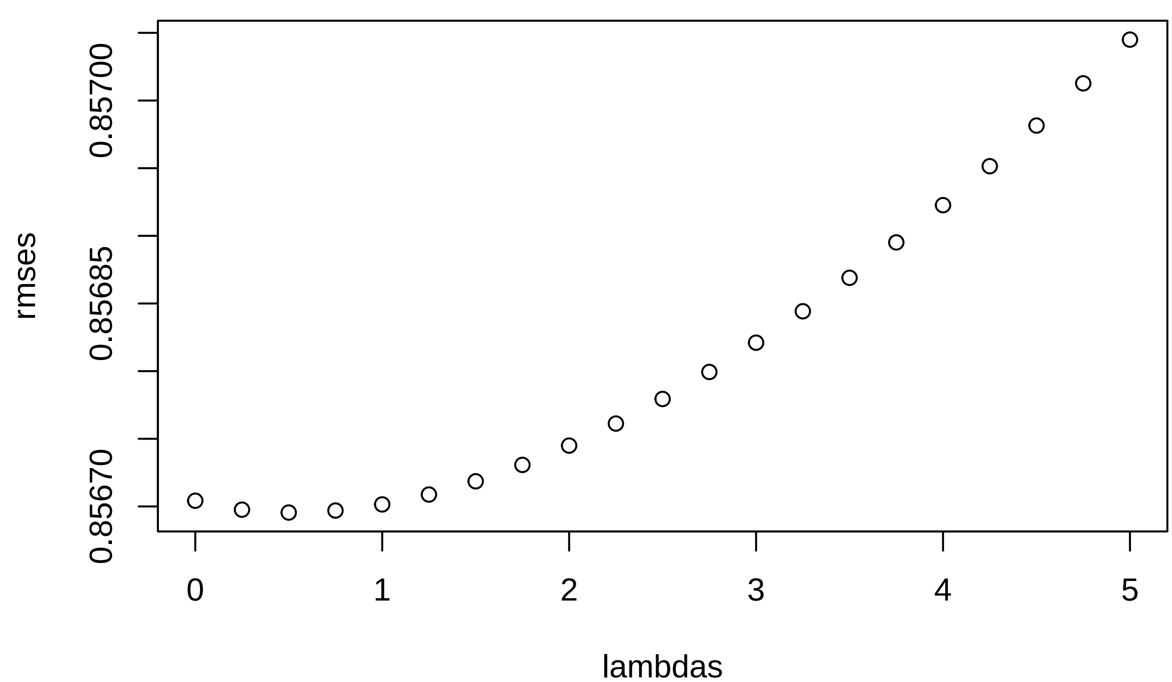
ii. Regularization

Regularization in regression is a technique used to prevent overfitting in a model by adding a penalty term to the loss function. The penalty term reduces the magnitude of the coefficients of the predictors, which makes the model more robust to the presence of outliers and noisy data. Lambda ( ) is a tuning parameter used in regularization to control the strength of the penalty term. The value of lambda determines the amount of regularization applied to the model. A higher value of lambda leads to more regularization and a simpler model, while a lower value of lambda leads to less regularization and a more complex model. By adjusting the value of lambda, it is possible to find the best balance between underfitting and overfitting in the model.

```
#tuning parameters
lambdas <- seq(0, 5, 0.25)

#RMSES Table with each tuning parameter
rmses <- sapply(lambdas, function(l){

  #Mean Ratings
  mu_reg <- mean(df$rating)

  #beta based on movieID
```

```
  b_i_reg <- df %>%
    group_by(movieId) %>%
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))

  #beta based on userid and movieId
  b_u_reg <- df %>%
    left_join(b_i_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))

  #predicting the model based on calculated betas.
  predicted_ratings_b_i_u <-
    df %>%
    left_join(b_i_reg, by = "movieId") %>%
    left_join(b_u_reg, by = "userId") %>%
    mutate(pred = mu_reg + b_i_reg + b_u_reg) %>%
    .$pred


  return(RMSE(predicted_ratings_b_i_u, edx$rating))
    })
```

```
plot(y =rmses, x =lambdas)
```



iii. Performance

```r
#Tuning parameters and RMSE table

tpr <- data.frame(Lambdas = lambdas,
                  RMSE = rmses)

tpr[which.min(tpr$RMSE),]
```

```
##   Lambdas      RMSE
## 3     0.5 0.8566955
```

6) Conclusion

RMSE was achieved through simple regression and it was further reduced and regularised by tuning parameters. It was much below 0.9 to prevent overfitting in the model. This simple model shows how powerful a simple regression can be and how it can predict ratings to such an extend.