## PEDIREDLA SUHAAS

## MovieLens Project

## HarvardX: PH125.9x - Capstone Project

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

# MovieLens Rating Prediction Project Code

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

#### Introduction ####

## Dataset ##

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

# Create edx set, validation set, and submission file

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

# Note: this process could take a couple of minutes for loading required package: tidyverse and package caret

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")

dl <- tempfile()

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

ratings <- read.table(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")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],

title = as.character(title),

genres = as.character(genres))

movielens <- left\_join(ratings, movies, by = "movieId")

# The Validation subset will be 10% of the MovieLens data.

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 subset:

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)

#### Methods and Analysis ####

### Data Analysis ###

# Head

head(edx) %>%

print.data.frame()

# Total unique movies and users

summary(edx)

# Number of unique movies and users in the edx dataset

edx %>%

summarize(n\_users = n\_distinct(userId),

n\_movies = n\_distinct(movieId))

# Ratings distribution

edx %>%

ggplot(aes(rating)) +

geom\_histogram(binwidth = 0.25, color = "black") +

scale\_x\_discrete(limits = c(seq(0.5,5,0.5))) +

scale\_y\_continuous(breaks = c(seq(0, 3000000, 500000))) +

ggtitle("Rating distribution")

# Plot number of ratings per movie

edx %>%

count(movieId) %>%

ggplot(aes(n)) +

geom\_histogram(bins = 30, color = "black") +

scale\_x\_log10() +

xlab("Number of ratings") +

ylab("Number of movies") +

ggtitle("Number of ratings per movie")

# Table 20 movies rated only once

edx %>%

group\_by(movieId) %>%

summarize(count = n()) %>%

filter(count == 1) %>%

left\_join(edx, by = "movieId") %>%

group\_by(title) %>%

summarize(rating = rating, n\_rating = count) %>%

slice(1:20) %>%

knitr::kable()

# Plot number of ratings given by users

edx %>%

count(userId) %>%

ggplot(aes(n)) +

geom\_histogram(bins = 30, color = "black") +

scale\_x\_log10() +

xlab("Number of ratings") +

ylab("Number of users") +

ggtitle("Number of ratings given by users")

# Plot mean movie ratings given by users

edx %>%

group\_by(userId) %>%

filter(n() >= 100) %>%

summarize(b\_u = mean(rating)) %>%

ggplot(aes(b\_u)) +

geom\_histogram(bins = 30, color = "black") +

xlab("Mean rating") +

ylab("Number of users") +

ggtitle("Mean movie ratings given by users") +

scale\_x\_discrete(limits = c(seq(0.5,5,0.5))) +

theme\_light()

### Modelling Approach ###

## Average movie rating model ##

# Compute the dataset's mean rating

mu <- mean(edx$rating)

mu

# Test results based on simple prediction

naive\_rmse <- RMSE(validation$rating, mu)

naive\_rmse

# Check results

# Save prediction in data frame

rmse\_results <- data\_frame(method = "Average movie rating model", RMSE = naive\_rmse)

rmse\_results %>% knitr::kable()

## Movie effect model ##

# Simple model taking into account the movie effect b\_i

# Subtract the rating minus the mean for each rating the movie received

# Plot number of movies with the computed b\_i

movie\_avgs <- edx %>%

group\_by(movieId) %>%

summarize(b\_i = mean(rating - mu))

movie\_avgs %>% qplot(b\_i, geom ="histogram", bins = 10, data = ., color = I("black"),

ylab = "Number of movies", main = "Number of movies with the computed b\_i")

# Test and save rmse results

predicted\_ratings <- mu + validation %>%

left\_join(movie\_avgs, by='movieId') %>%

pull(b\_i)

model\_1\_rmse <- RMSE(predicted\_ratings, validation$rating)

rmse\_results <- bind\_rows(rmse\_results,

data\_frame(method="Movie effect model",

RMSE = model\_1\_rmse ))

# Check results

rmse\_results %>% knitr::kable()

## Movie and user effect model ##

# Plot penaly term user effect #

user\_avgs<- edx %>%

left\_join(movie\_avgs, by='movieId') %>%

group\_by(userId) %>%

filter(n() >= 100) %>%

summarize(b\_u = mean(rating - mu - b\_i))

user\_avgs%>% qplot(b\_u, geom ="histogram", bins = 30, data = ., color = I("black"))

user\_avgs <- edx %>%

left\_join(movie\_avgs, by='movieId') %>%

group\_by(userId) %>%

summarize(b\_u = mean(rating - mu - b\_i))

# Test and save rmse results

predicted\_ratings <- validation%>%

left\_join(movie\_avgs, by='movieId') %>%

left\_join(user\_avgs, by='userId') %>%

mutate(pred = mu + b\_i + b\_u) %>%

pull(pred)

model\_2\_rmse <- RMSE(predicted\_ratings, validation$rating)

rmse\_results <- bind\_rows(rmse\_results,

data\_frame(method="Movie and user effect model",

RMSE = model\_2\_rmse))

# Check result

rmse\_results %>% knitr::kable()

## Regularized movie and user effect model ##

# lambda is a tuning parameter

# Use cross-validation to choose it.

lambdas <- seq(0, 10, 0.25)

# For each lambda,find b\_i & b\_u, followed by rating prediction & testing

# note:the below code could take some time

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))

})

# Plot rmses vs lambdas to select the optimal lambda

qplot(lambdas, rmses)

# The optimal lambda

lambda <- lambdas[which.min(rmses)]

lambda

# Test and save results

rmse\_results <- bind\_rows(rmse\_results,

data\_frame(method="Regularized movie and user effect model",

RMSE = min(rmses)))

# Check result

rmse\_results %>% knitr::kable()

#### Results ####

# RMSE results overview

rmse\_results %>% knitr::kable()

#### Appendix ####

print("Operating System:")

version