MovieLens Project

HarvardX: PH125.9x Data Science: Capstone: Sudha Kankipati

2020-06-10

Introduction

Recommendation systems use rating data from many products and users to make recommendations for a specific user.
We Will use the following code to generate datasets and develop algorithm using the edx set. For a final test of algorithm, We predict movie ratings in the validation set as if they were unknown.RMSE will be used to evaluate how close predictions are to the true values in the validation set.

Data Preparation

creating edx set and validation set Note: this process could take a couple of minutes

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

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

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1.3.0
```

```
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------ tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

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

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

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

```
## Loading required package: data.table
```

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

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

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

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")
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")
```

Validation set will be 10% of MovieLens data

```
set.seed(1, sample.kind="Rounding")
```

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

if using R 3.5 or earlier, use `set.seed(1)` instead

```
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)
```

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

```
edx <- rbind(edx, removed)

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

Data Exploration

Before creating the model,Understanding the features of the rating data set. This step will help build a better model.

*using dataMaid code in R script

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

library(dataMaid)

makeDataReport(edx, replace=TRUE)

- Note I converted R code to .Rmd code to incorporate the results in PDF document Function call: makeDataReport(data = edx, replace = TRUE)

Data report overview

The dataset examined has the following dimensions:

Feature	Result
Number of observations	9000055
Number of variables	6

Checks performed

The following variable checks were performed, depending on the data type of each variable:

	character	factor	labelled	haven labelled	numeric	integer	logical	Date
Identify miscoded missing values	×	×	×	×	×	×		×
Identify prefixed and suffixed whitespace	×	×	×	×				
Identify levels with < 6 obs.	×	×	×	×				
Identify case issues	×	×	×	×				
Identify misclassified numeric or integer variables	×	×	×	×				
Identify outliers					×	×		×

Please note that all numerical values in the following have been rounded to 2 decimals.

Summary table

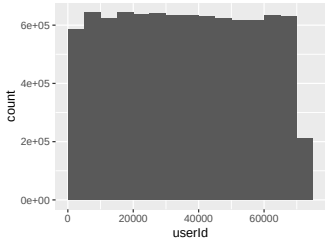
	Variable class	# unique values	Missing observations	Any problems?
userId	integer	69878	0.00 %	
movieId	numeric	10677	0.00 %	×

	Variable class	# unique values	Missing observations	Any problems?
rating	numeric	10	0.00 %	×
timestamp	integer	6519590	0.00 %	
title	character	1	100.00 %	×
genres	character	1	100.00 %	×

Variable list

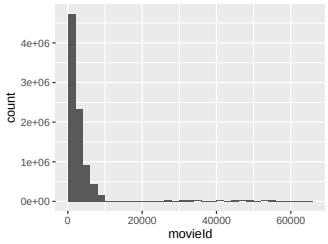
userId

Feature	Result
Variable type	integer
Number of missing obs.	0 (0 %)
Number of unique values	69878
Median	35738
1st and 3rd quartiles	18124; 53607
Min. and max.	1; 71567



movieId

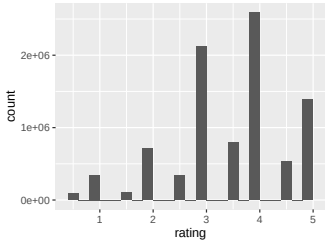
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	10677
Median	1834
1st and 3rd quartiles	648; 3626
Min. and max.	1; 65133



- Note that the following possible outlier values were detected: "25736", "25737", "25744", "25746", "25750", "25752", "25753", "25755", "25757", "25759" (2351 additional values omitted).

rating

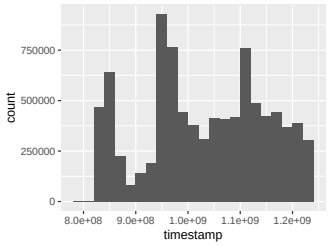
Feature	Result
Variable type	numeric
Number of missing obs.	0 (0 %)
Number of unique values	10
Median	4
1st and 3rd quartiles	3; 4
Min. and max.	0.5; 5



- Note that the following possible outlier values were detected: "4.5", "5".

timestamp

Feature	Result
Variable type	integer
Number of missing obs.	0 (0 %)
Number of unique values	6519590
Median	1035493918
1st and 3rd quartiles	946768283; 1126750881
Min. and max.	789652009; 1231131736



---

title

- The variable only takes one value: "NA".

---

genres

- The variable only takes one value: "NA".

---

showing train set

```
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title genres
##     <int>   <dbl>  <dbl>     <int> <chr> <chr>
##  1      1     122      5 838985046 <NA>  <NA>
##  2      1     185      5 838983525 <NA>  <NA>
##  3      1     292      5 838983421 <NA>  <NA>
##  4      1     316      5 838983392 <NA>  <NA>
##  5      1     329      5 838983392 <NA>  <NA>
##  6      1     355      5 838984474 <NA>  <NA>
##  7      1     356      5 838983653 <NA>  <NA>
##  8      1     362      5 838984885 <NA>  <NA>
##  9      1     364      5 838983707 <NA>  <NA>
## 10      1     370      5 838984596 <NA>  <NA>
## # ... with 9,000,045 more rows
```

```
edx %>% summarize(
  n_users=n_distinct(userId),# unique users from train set
  n_movies=n_distinct(movieId),# unique movies from train set
  min_rating=min(rating), # the lowest rating in train set
  max_rating=max(rating) # the highest rating in train set
)
```

```
##   n_users n_movies min_rating max_rating
## 1   69878    10677        0.5          5
```

Data Cleaning

Removing genres and timestamp.

```
edx <- edx %>% select(userId, movieId, rating, title)
validation <- validation %>% select(userId, movieId, rating, title)
```

Data Visualization

Movies exploration

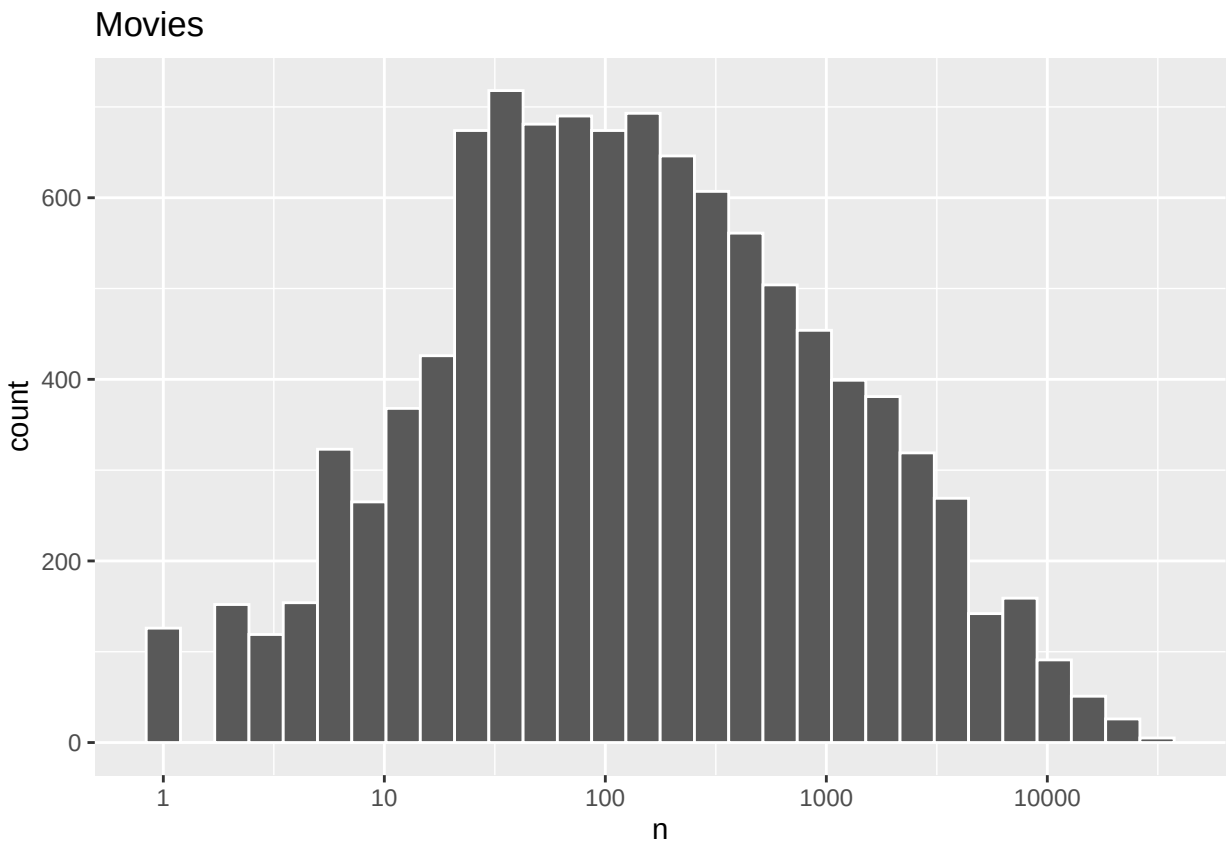
Checking number of different movies present in 'edx' set

```
length(unique(edx$movieId))
```

```
## [1] 10677
```

Distribution of movies: Movies rated more than others (histogram)

```
edx %>% group_by(movieId) %>%
  summarise(n=n()) %>%
  ggplot(aes(n)) +
  geom_histogram(bin = 30, color = "white") +
  scale_x_log10() +
  ggtitle("Movies")
```



Users exploration

Checking number of different users present in 'edx' set

```
length(unique(edx$userId))
```

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

Distribution of users

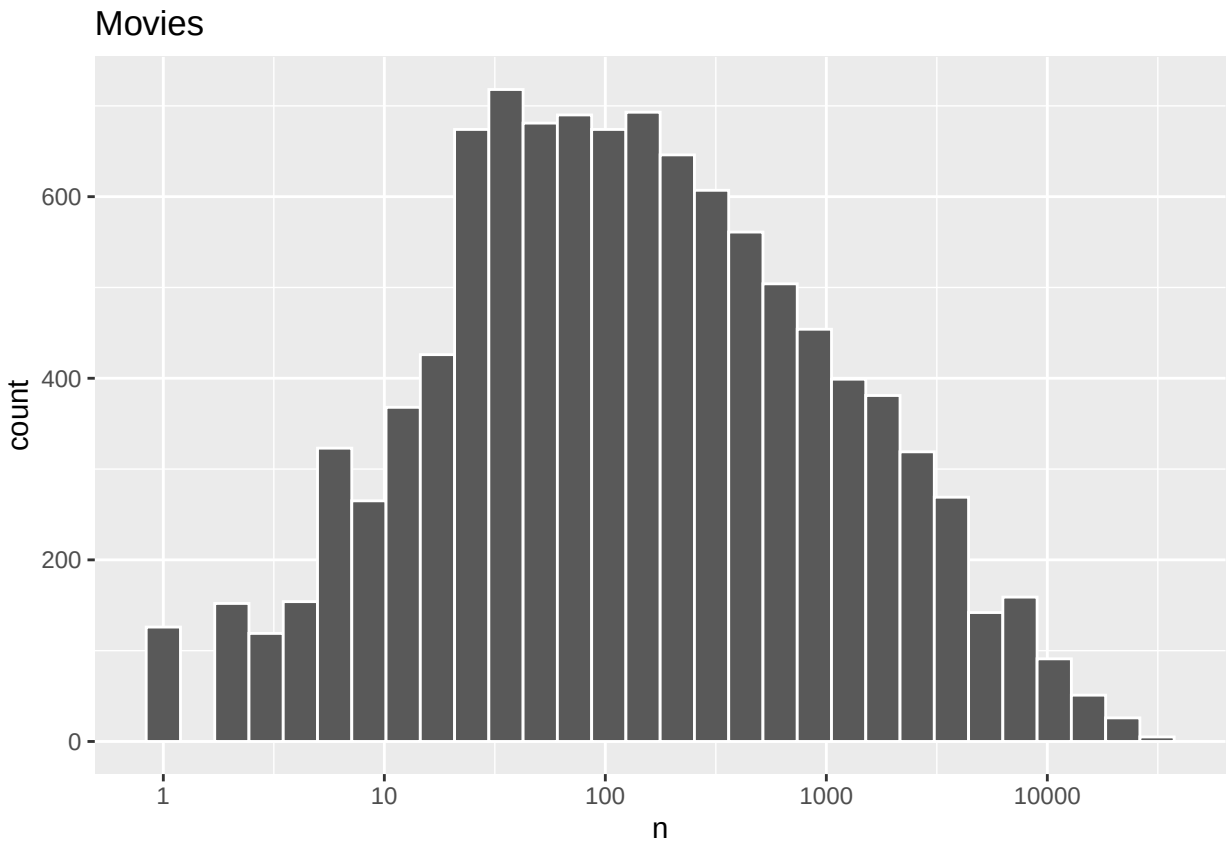
```
edx %>% group_by(userId) %>%
  summarise(n=n()) %>%
  arrange(n) %>%
  head()
```

```
## # A tibble: 6 x 2
##   userId     n
##    <int> <int>
## 1  62516    10
## 2  22170    12
## 3  15719    13
## 4  50608    13
## 5    901    14
## 6   1833    14
```

Insights

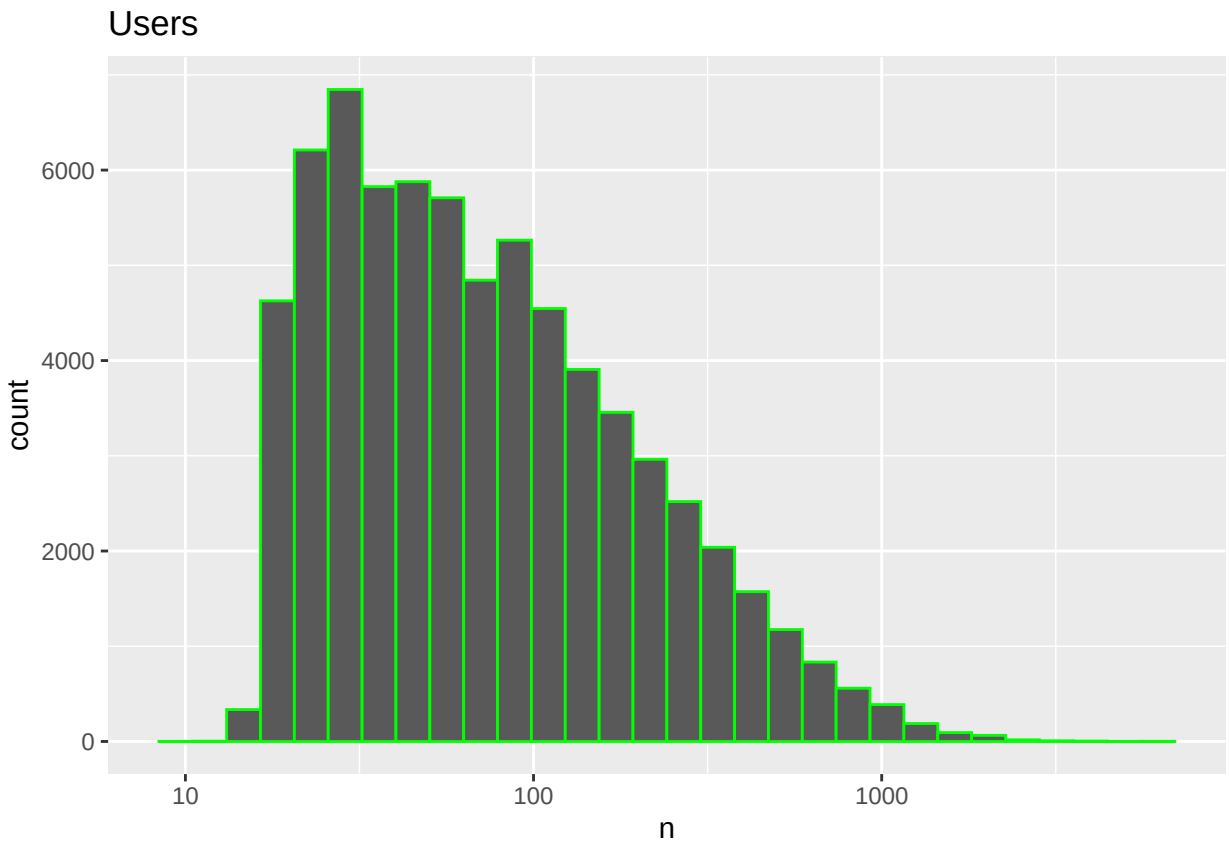
- The first thing we notice is that some movies get rated more than others. Here is the distribution: plot count rating by movie

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "White") +
  scale_x_log10() +
  ggtitle("Movies")
```



* Our second observation is that some users are more active than others at rating movies: plot count rating by user

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "green") +
  scale_x_log10() +
  ggtitle("Users")
```


Users
6000
4000
2000
0
count
10
100
1000
n

Methods and Analysis

RMSE

Root Mean Squared Error (RMSE) is the indicator used to compare the predicted value with the actual outcome. During the model development, we use the test(edx) set to predict the outcome. When the model is ready, then we use the 'validation' set.

Linear Model

Average movie rating model

- compute mean rating

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

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

- compute root mean standard error

```
naive_rmse <- RMSE(edx$rating, mu)
```

- Check results and Save prediction in tibble

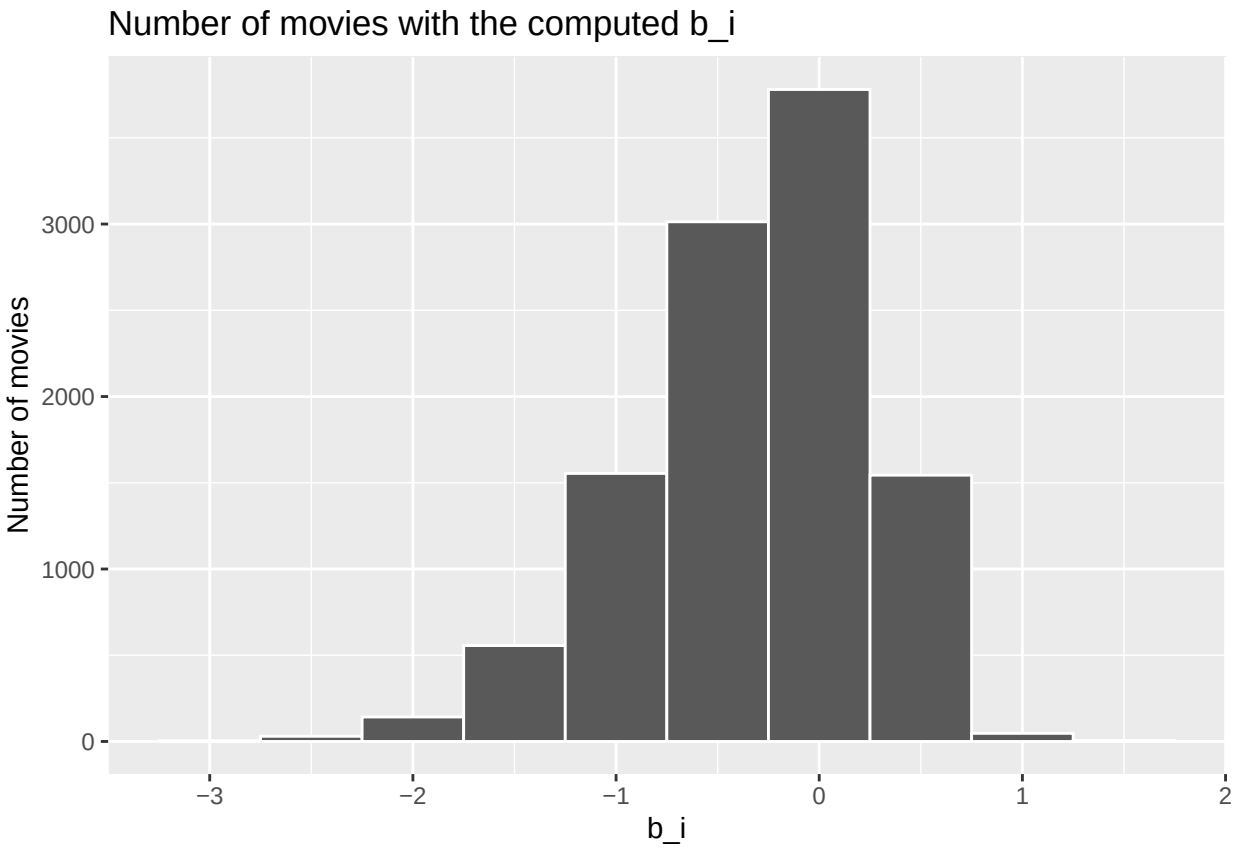
```
rmse_results <- tibble(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.060331

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("White"),
         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,
                          tibble(method="Movie effect model",
                                     RMSE = model_1_rmse ))
```

- Check results

```
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie effect model	0.9439087

Matrix factorization

recosystem is a package for recommendation system using Matrix Factorization. High performance multi-core parallel computing is supported in this package.

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

```
## Loading required package: recosystem
```

```
set.seed(1)
```

This is a randomized algorithm

Converting the train and test sets into recosystem input format

```
train_data <-  with(edx, data_memory(user_index = userId,
                                     item_index = movieId,
                                     rating     = rating))
test_data  <-  with(validation,  data_memory(user_index = userId,
                                     item_index = movieId,
                                     rating     = rating))
```

creating the model object

```
recommender <-  recosystem::Reco()
```

Select the best tuning parameters

```
opts <- recommender$tune(train_data, opts = list(dim = c(10, 20, 30),
                                       lrate = c(0.1, 0.2),
                                       costp_l2 = c(0.01, 0.1),
                                       costq_l2 = c(0.01, 0.1),
                                       nthread  = 4, niter = 10))
```

Training the algorithm

```
recommender$train(train_data, opts = c(opts$min, nthread = 4, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9707   1.1985e+07
##    1       0.8728   9.8895e+06
##    2       0.8390   9.1750e+06
##    3       0.8170   8.7524e+06
##    4       0.8015   8.4747e+06
##    5       0.7898   8.2763e+06
##    6       0.7801   8.1299e+06
##    7       0.7720   8.0076e+06
##    8       0.7651   7.9105e+06
##    9       0.7589   7.8277e+06
##   10       0.7537   7.7597e+06
##   11       0.7489   7.7006e+06
##   12       0.7446   7.6497e+06
##   13       0.7408   7.6045e+06
##   14       0.7372   7.5640e+06
##   15       0.7341   7.5298e+06
##   16       0.7311   7.4999e+06
##   17       0.7284   7.4689e+06
##   18       0.7259   7.4457e+06
##   19       0.7236   7.4217e+06
```

Calculate the predicted values

- We predict unknown entries in the rating matrix on test set. use the `$predict()` method to compute predicted values return predicted values in memory

```
predicted_ratings <-  recommender$predict(test_data, out_memory())
head(predicted_ratings, 10)#We use the test set for the final assessment
```

```
##  [1] 4.152528 5.054093 4.684475 3.453819 4.479469 2.742820 3.977404 4.414435
##  [9] 4.387129 3.430398
```

- ceiling rating at 5

```
ind <- which(predicted_ratings > 5)
predicted_ratings[ind] <- 5
```

- floor rating at 0.50

```
ind <- which(predicted_ratings < 0.5)
predicted_ratings[ind] <- 0.5
```

- creating a results table with this approach

```
model_2_rmse <- RMSE(validation$rating, predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie and User effects and Matrix Fact. on test set",
                                 RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie effect model	0.9439087
Movie and User effects and Matrix Fact. on test set	0.7821349

```
set.seed(1)
```

Converting 'edx' and 'validation' sets to recosystem input format

```
edx_reco <-  with(edx, data_memory(user_index = userId,
                                   item_index = movieId,
                                   rating = rating))
validation_reco  <-  with(validation, data_memory(user_index = userId,
                                                  item_index = movieId,
                                                  rating = rating))
```

creating the model object

```
recommender <-  recosystem::Reco()
```

Tune the parameters

```
opts <-  recommender$tune(edx_reco, opts = list(dim = c(10, 20, 30),
                                                lrate = c(0.1, 0.2),
                                                costp_l2 = c(0.01, 0.1),
                                                costq_l2 = c(0.01, 0.1),
                                                nthread  = 4, niter = 10))
```

Training the model

```
recommender$train(edx_reco, opts = c(opts$min, nthread = 4, niter = 20))
```

```
## iter      tr_rmse          obj
##    0       0.9691   1.1968e+07
```

```
##    1       0.8726   9.8818e+06
##    2       0.8394   9.1781e+06
##    3       0.8174   8.7609e+06
##    4       0.8015   8.4766e+06
##    5       0.7897   8.2783e+06
##    6       0.7798   8.1272e+06
##    7       0.7716   8.0042e+06
##    8       0.7647   7.9060e+06
##    9       0.7586   7.8254e+06
##   10       0.7534   7.7552e+06
##   11       0.7488   7.6999e+06
##   12       0.7445   7.6481e+06
##   13       0.7407   7.6035e+06
##   14       0.7373   7.5663e+06
##   15       0.7340   7.5302e+06
##   16       0.7312   7.4989e+06
##   17       0.7285   7.4707e+06
##   18       0.7261   7.4457e+06
##   19       0.7238   7.4229e+06
```

- creating a results table with this and prior approaches Calculate the prediction

```
predicted_ratings <-  recommender$predict(validation_reco, out_memory())


model_3_rmse <- RMSE(validation$rating, predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie and User effects and Matrix Fact. on validation set",
                                 RMSE = model_3_rmse))
```

Results

- Results of all Models I tried in the process to get less RMSE

```
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie effect model	0.9439087
Movie and User effects and Matrix Fact. on test set	0.7821349
Movie and User effects and Matrix Fact. on validation set	0.7825573

Conclusion

We reach a RMSE of 0.7825573 on validation set using the full edx set for training. The model Movie and User effects and Matrix Factorization achieved this performance using the methodology presented in the course's book (https://rafalab.github.io/dsbook/largedatasets.html#recommendation-systems)

Report generation information:

- created by: Sudha Kankipati
- Report creation time: Wed Jun 10 2020
- R version 4.0.0 (2020-04-24).
- Platform: x86_64-w64-mingw32/x64 (64-bit)(Windows 10 x64 (build 18363)).
- Placed files for this project in https://github.com/DrSudhaK/MovieLensProject.git