Capstone Project

Lewis Rincon Castano

2024-06-24

Introduction

My HarvardX Capstone Project focuses on the MovieLens dataset, providing an in-depth analysis and insights into movie preferences and ratings. The project's aim to analyze movie preferences and ratings using the MovieLens dataset1. It emphasizes the use of statistical techniques and visualizations to gain insights into user and movie rating patterns. The report details the data preprocessing and exploration process, including creating a rating matrix and examining rating patterns.

Methods/analysiS

The project involves data preprocessing and exploration, including creating a rating matrix and examining user and movie rating patterns. It utilizes a variety of R libraries for data manipulation, visualization, and machine learning. We have six columns with 8,100,050 rows, where a number of unique users: 69878 and unique movies: 10667. The summary(train_set$rating) function provides a statistical summary of the rating column in the train_set dataframe. Here are the key statistics:

- Minimum: The lowest rating is 0.5.
- 1st Quartile: 25% of the ratings are below 3.0.
- Median: The middle value of the ratings is 4.0.
- Mean: The average rating is approximately 3.512.
- 3rd Quartile: 75% of the ratings are below 4.0.
- Maximum: The highest rating is 5.0.

These statistics give a quick overview of the distribution of movie ratings in the dataset. Rating Distribution Graph: This graph is a histogram that displays the frequency of movie ratings in the dataset.The x-axis represents the rating value, and the y-axis shows the count of ratings.The graph is described as left-skewed, indicating that most ratings are above three stars, suggesting a generally positive reception of movies in the dataset.

Average Rating Graphs: There are two histograms, one for the average rating per movie and another for the average rating per user. Both graphs use a bin width of 0.1 to group the average ratings. The x-axis represents the average rating, and the y-axis indicates the count of movies or users that fall into each average rating category. These graphs help to visualize the central tendency and dispersion of average ratings among movies and users.

Results

The RMSE (Root Mean Square Error) scores from the Capstone Project analysis of the MovieLens dataset are as follows:

- Mean-Based Model: The RMSE for the mean-based model on the validation set is 0.9439798279509581.
- Mean Rating Model: The RMSE for the mean rating model on the final hold-out test set is 1.0612022.
- Linear Regression Model: The RMSE for the linear regression model on the final hold-out test set is 1.061177.
- XGBoost Model: The RMSE for the XGBoost model on the final hold-out test set is 1.0287993.
- Bias Subtraction Model: The RMSE for the bias subtraction model on the final hold-out test set is 0.89142214.

The lowest RMSE score is from the Bias Subtraction Model, which is 0.8914221. This indicates that the Bias Subtraction Model has the best predictive accuracy among the models evaluated.

Conclusion The project concludes with the evaluation of different models using

RMSE (Root Mean Square Error) to determine their accuracy in predicting movie ratings, highlighting the performance of the bias subtraction model.

The project involves complex models and large datasets, which are computationally demanding and can lead to system crashes on personal laptops with limited resources like 8GB RAM.

Load libraries

```
## -- Attaching core tidyverse packages ------------------------ tidyverse
2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
##
## Attaching package: 'data.table'
##
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
##
## The following object is masked from 'package:purrr':
##
##     transpose
##
##
## Loading required package: lattice
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
##
## Attaching package: 'scales'
##
##
## The following object is masked from 'package:purrr':
##
##     discard
##
##
## The following object is masked from 'package:readr':
##
##     col_factor
##
##
## Loading required package: Matrix
```

```
##
##
## Attaching package: 'Matrix'
##
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Loading required package: arules
##
##
## Attaching package: 'arules'
##
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
##
##
## Loading required package: proxy
##
##
## Attaching package: 'proxy'
##
##
## The following object is masked from 'package:Matrix':
##
##     as.matrix
##
##
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
##
##
## The following object is masked from 'package:base':
##
##     as.matrix
##
##
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
```

```
##   print.registry_entry proxy
##
##
## Attaching package: 'recommenderlab'
##
##
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
##
##
##
## Attaching package: 'xgboost'
##
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

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

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

# Set the timeout option to 120 seconds to allow for large file downloads
options(timeout = 120)

# Create a temporary file for the download
dl <- tempfile()

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

# Define the paths for the ratings and movies files
ratings_file <- "ml-10M100K/ratings.dat"
movies_file <- "ml-10M100K/movies.dat"

# Unzip the ratings file if it doesn't already exist
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

# Unzip the movies file if it doesn't already exist
if(!file.exists(movies_file))
  unzip(dl, movies_file)

# Read the ratings data and split into columns
```

```r
ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"),
simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")

# Convert columns to appropriate data types
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

# Read the movies data and split into columns
movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"),
simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")

# Convert columns to appropriate data types
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

# Merge the ratings and movies dataframes by movieId
movielens <- left_join(ratings, movies, by = "movieId")

# Set a random seed for reproducibility and create a test set (10% of data)
set.seed(1, sample.kind="Rounding") # Use this if using R 3.6 or later
# set.seed(1) # Use this 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,]

# Ensure userId and movieId in the final hold-out test set are also in the
edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

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

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

edx <- rbind(edx, removed)

# Clean up the workspace by removing unnecessary objects
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

```r
# Split edx data into training and validation sets
set.seed(1, sample.kind = "Rounding")
train_index <- createDataPartition(y = edx$rating, times = 1, p = 0.9, list =
FALSE)
train_set <- edx[train_index, ]
validation_set <- edx[-train_index, ]

# Check the structure of the training set
glimpse(train_set)
```

```
## Rows: 8,100,050
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2, ~
## $ movieId   <int> 122, 185, 292, 316, 329, 355, 356, 364, 370, 377, 420,
466, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392,
83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak
(1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller",
"Action|Drama|Sci~
```

Summary statistics of the ratings

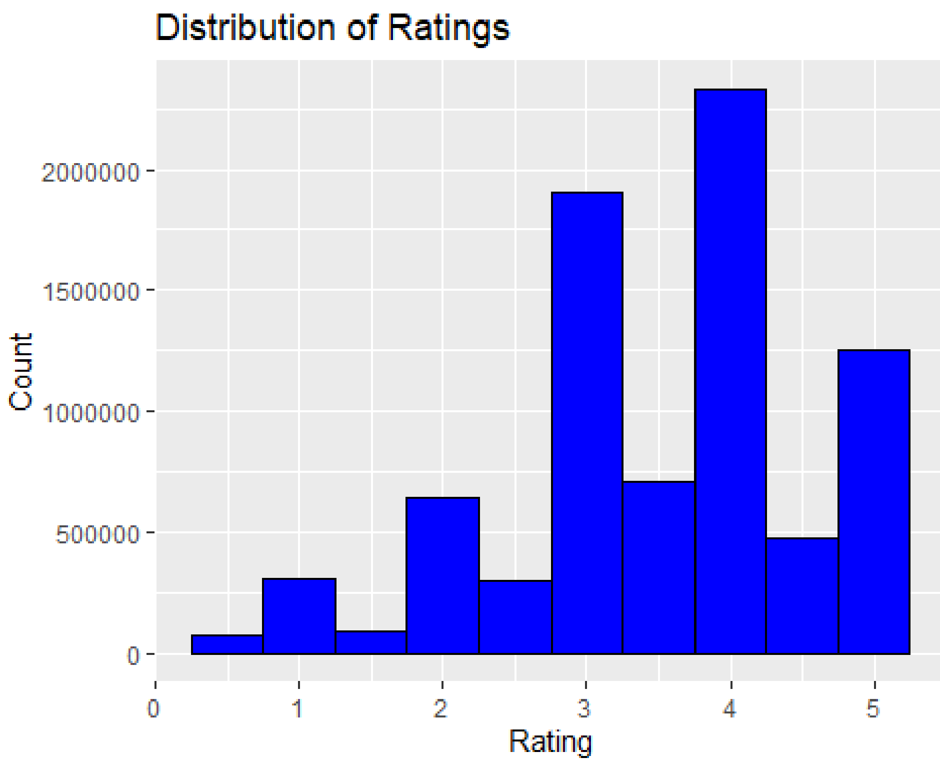
```r
# Summary statistics of the ratings
summary(train_set$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

Distribution of ratings:

On the below bar chart, we identify a left skewed graph where most of the ratings counts are above three stars.

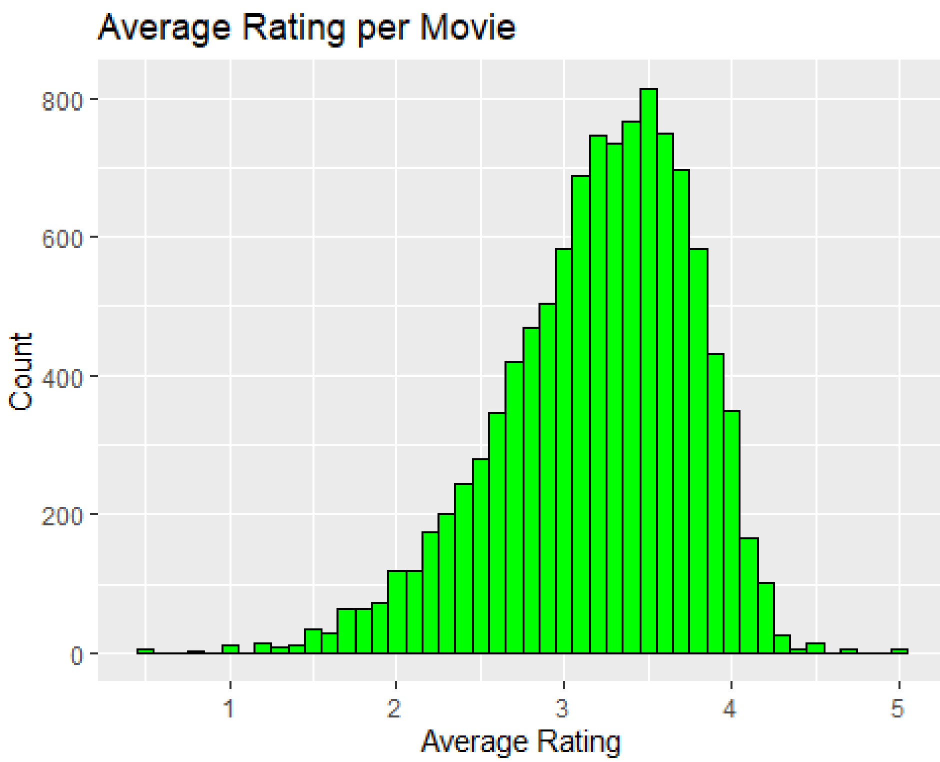
```r
# Distribution of ratings
ggplot(train_set, aes(x = rating)) +
  geom_histogram(binwidth = 0.5, fill = "blue", color = "black") +
  ggtitle("Distribution of Ratings") +
  xlab("Rating") +
  ylab("Count")
```



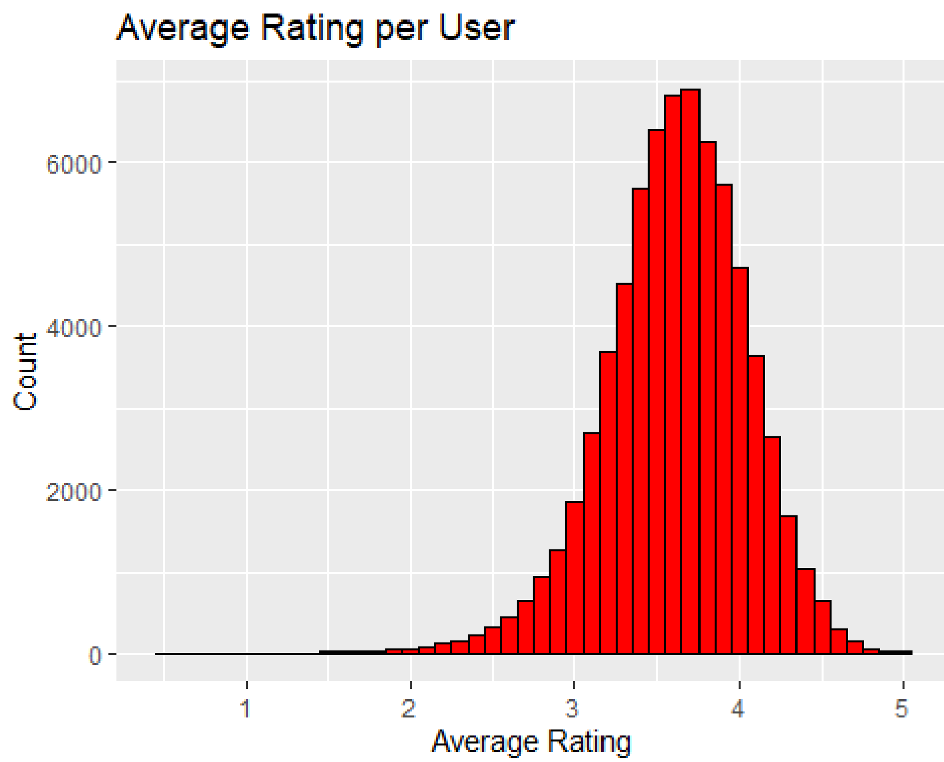
Number of unique users and movies

```
# Number of unique users and movies
num_users <- n_distinct(train_set$userId)
num_movies <- n_distinct(train_set$movieId)
cat("Number of unique users:", num_users, "\n")

## Number of unique users: 69878

cat("Number of unique movies:", num_movies, "\n")

## Number of unique movies: 10667

# Average rating per movie
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(avg_rating = mean(rating))


# Average rating per user
user_avgs <- train_set %>%
  group_by(userId) %>%
  summarize(avg_rating = mean(rating))

# Plot average ratings per movie
ggplot(movie_avgs, aes(x = avg_rating)) +
  geom_histogram(binwidth = 0.1, fill = "green", color = "black") +
  ggtitle("Average Rating per Movie") +
```

```
  xlab("Average Rating") +
  ylab("Count")
```




```
# Plot average ratings per user
ggplot(user_avgs, aes(x = avg_rating)) +
  geom_histogram(binwidth = 0.1, fill = "red", color = "black") +
  ggtitle("Average Rating per User") +
  xlab("Average Rating") +
  ylab("Count")
```




```
# Calculate mean ratings for each movie
mean_ratings <- train_set %>%
  group_by(movieId) %>%
  summarise(mean_rating = mean(rating, na.rm = TRUE))

# Merge mean ratings with validation_set to predict ratings
predictions_mean <- merge(validation_set, mean_ratings, by = "movieId", all.x
= TRUE)

# Calculate RMSE for mean-based model
rmse_mean <- sqrt(mean((predictions_mean$rating -
predictions_mean$mean_rating)^2, na.rm = TRUE))
print(paste("RMSE for Mean-Based Model on validation_set:", rmse_mean))

## [1] "RMSE for Mean-Based Model on validation_set: 0.943979827950958"

# Model Mean Rating
mean_rating <- mean(train_set$rating)

# Predict ratings using the mean rating
predictions_mean <- rep(mean_rating, nrow(validation_set))

# Calculate RMSE for the mean rating model on validation set
rmse_mean <- RMSE(predictions_mean, validation_set$rating)
cat("RMSE for mean rating model (validation set):", rmse_mean, "\n")

## RMSE for mean rating model (validation set): 1.059799
```

```
# Predict ratings using the mean rating for final hold-out test set
predictions_mean_holdout <- rep(mean_rating, nrow(final_holdout_test))

# Calculate RMSE for the mean rating model on final hold-out test set
rmse_mean_holdout <- RMSE(predictions_mean_holdout,
final_holdout_test$rating)
cat("RMSE for mean rating model (final hold-out test set):",
rmse_mean_holdout, "\n")

## RMSE for mean rating model (final hold-out test set): 1.061202

# Model - Linear Regression
lm_model <- lm(rating ~ movieId + userId, data = train_set)

# Predict ratings using the linear regression model on validation set
predictions_lm <- predict(lm_model, validation_set)

# Calculate RMSE for the linear regression model on validation set
rmse_lm <- RMSE(predictions_lm, validation_set$rating)
cat("RMSE for linear regression model (validation set):", rmse_lm, "\n")

## RMSE for linear regression model (validation set): 1.059764

# Predict ratings using the linear regression model on final hold-out test
set
predictions_lm_holdout <- predict(lm_model, final_holdout_test)

# Calculate RMSE for the linear regression model on final hold-out test set
rmse_lm_holdout <- RMSE(predictions_lm_holdout, final_holdout_test$rating)
cat("RMSE for linear regression model (final hold-out test set):",
rmse_lm_holdout, "\n")

## RMSE for linear regression model (final hold-out test set): 1.061177

# Model - Prepare data for xgboost
train_matrix <- xgb.DMatrix(data = as.matrix(train_set %>% select(userId,
movieId)),
                            label = train_set$rating)
validation_matrix <- xgb.DMatrix(data = as.matrix(validation_set %>%
select(userId, movieId)),
                                  label = validation_set$rating)
final_holdout_matrix <- xgb.DMatrix(data = as.matrix(final_holdout_test %>%
select(userId, movieId)),
                                     label = final_holdout_test$rating)

# Set parameters for xgboost
params <- list(
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 5,
  subsample = 0.8,
```

```r
   colsample_bytree = 0.8
)

 # Train the model
set.seed(1)
xgb_model <- xgboost(data = train_matrix, params = params, nrounds = 100,
verbose = 0)

# Predict and calculate RMSE for validation set
predictions_xgb <- predict(xgb_model, validation_matrix)
rmse_xgb <- RMSE(predictions_xgb, validation_set$rating)
cat("RMSE for XGBoost model (validation set):", rmse_xgb, "\n")

## RMSE for XGBoost model (validation set): 1.027803

# Predict and calculate RMSE for final hold-out test set
predictions_xgb_holdout <- predict(xgb_model, final_holdout_matrix)
rmse_xgb_holdout <- RMSE(predictions_xgb_holdout, final_holdout_test$rating)
cat("RMSE for XGBoost model (final hold-out test set):", rmse_xgb_holdout,
"\n")

## RMSE for XGBoost model (final hold-out test set): 1.028799

# Model -  Calculate global mean rating
global_mean <- mean(train_set$rating)

# Calculate user biases
user_biases <- train_set %>%
  group_by(userId) %>%
  summarize(user_bias = mean(rating - global_mean))

# Calculate item biases
item_biases <- train_set %>%
  group_by(movieId) %>%
  summarize(item_bias = mean(rating - global_mean - user_biases$user_bias))

# Predict ratings on validation set using bias subtraction
predictions_bias <- validation_set %>%
  left_join(user_biases, by = "userId") %>%
  left_join(item_biases, by = "movieId") %>%
  mutate(predicted_rating = global_mean + user_bias + item_bias) %>%
  pull(predicted_rating)

# Calculate RMSE for bias subtraction model on validation set
rmse_bias <- RMSE(predictions_bias, validation_set$rating)
cat("RMSE for bias subtraction model (validation set):", rmse_bias, "\n")

## RMSE for bias subtraction model (validation set): 0.8909362

# Predict ratings for final hold-out test set using bias subtraction
predictions_bias_holdout <- final_holdout_test %>%
```

```r
  left_join(user_biases, by = "userId") %>%
  left_join(item_biases, by = "movieId") %>%
  mutate(predicted_rating = global_mean + user_bias + item_bias) %>%
  pull(predicted_rating)

# Calculate RMSE for bias subtraction model on final hold-out test set
rmse_bias_holdout <- RMSE(predictions_bias_holdout,
final_holdout_test$rating)
cat("RMSE for bias subtraction model (final hold-out test set):",
rmse_bias_holdout, "\n")

## RMSE for bias subtraction model (final hold-out test set): 0.8914221
```