Capstone: MovieLens Project Report

Michael G. Mielach

9 12 2022

Introduction/Overview/Executive Summary

[Describe the dataset and summarize the goal of the project and key steps that were performed]

A recommender system, also known as a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. The basic idea behind recommender systems is to find patterns in data and use them to predict users' interests in order to provide personalized recommendations. This is done by searching for patterns in data that can be used to predict which items a user might like, and then presenting those items to the user.

For the movielens project we use a 10% slice from the 10M version of the MovieLens dataset. The goal of this project is to develop a recommendation system using XXX. The system will be trained on inputs from the edx set to predict movie ratings in the validation set. The Root Mean Squared Error (RMSE) will be used to evaluate how closely the predictions match the true values in the validation set. The project will study various machine learning techniques, with a focus on regression models, ensemble methods (such as gradient boosting and random forest), and recommender engines (such as slopeOne and matrix factorization with gradient descent) to process and analyze the features.

Methods/Analysis

section that explains the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and your modeling approach

Dataset preparation:

The data set is loaded from the grouplens which split the data into edx set and 10% validation set. the edx set will be split into training and test set,and validation set will be used to final evaluation.

```
library(tidyverse)
library(caret)

options(timeout = 120)

#load data

    dl <- "ml-10M100K.zip"
    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)
```

```
# Create data frame

    ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),stringsA
    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")


# Create final hold-out/validation test set

    # Slice 10% out of MovieLens data

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

        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/validation 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/validation test set back into edx set

      removed <- anti_join(temp, final_holdout_test)
      edx <- rbind(edx, removed)

    # remove some items which are no longer required

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

    # Check for any NA value

      anyNA(edx)
```

Exploratory Data Analysis:

Quick summary of the dataset & validation set

Code:

```
summary(edx)
str(edx)

summary(final_holdout_test)
str(final_holdout_test)
```

Description: The edX dataset has 6 features and is made up of around 9,000,055 observations. The validation set, which is 10% of the 10M Movielens dataset, contains the same features as the edX set, but has 999,999 observations. The userId and movieId in the edX set are also present in the validation set. Each row in the dataset represents a rating given by a user to a movie. The "rating" column is the outcome that we want to predict.

Code:

```
edx %>% summarize(unique_users = length(unique(userId)),
                  unique_movies = length(unique(movieId)),
                  unique_genres = length(unique(genres)),
                  unique_rating = length(unique(rating)))
range(edx$rating)
```

Description: The dataset contains 69,878 unique users, 10,677 unique movies and 797 unique genres. Users can give ratings ranging from 0.5 to 5.0 being the highest resulting in 10 different rating scores.

Code:

```
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>%
  arrange(desc(count))

# histogram of ratings

edx %>%
  ggplot(aes(x= rating)) +
  geom_histogram(bins = 10, binwidth = 0.2) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  labs(x="rating category", y="ratings given", caption = "source data: edx set") +
  ggtitle("Count of ratings per category")

# histogram of number of ratings by movieId

edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, binwidth=0.2, color="black", ) +
  scale_x_log10() +
  labs(x="n", y="number of ratings", caption = "source data: edx set") +
  ggtitle("Number of ratings per movie")

# histogram of number of ratings by userId

edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, binwidth=0.2, color="black", ) +
  scale_x_log10() +
  labs(x="n", y="number of ratings", caption = "source data: edx set") +
  ggtitle("Number of ratings per user")
```

```
#inspection of titles

edx %>%
  group_by(title) %>%
  summarize(count=n(), rating = mean(rating)) %>%
  top_n(20,count) %>%
  arrange(desc(count))

#inspection of genre

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

edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Error bar plots by genres" , caption = "source data : edx set")


#inspection of time

edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Timestamp, time unit : week")+
  labs(subtitle = "average ratings",
       caption = "source data : edx set")
```

Description: The top 5 ratings, in order from most to least common, are: 4, 3, 5, 3.5, and 2. The histogram indicates that ratings with half stars are less frequent than ratings with whole stars. Exploring the number of ratings by movieId and by userId through visualization shows that some movies receive more ratings than others, and some users are more active in rating movies. This likely explains the presence of movie effects and user effects in the data. Looking at the genre data the most frequently rated are Drama, Comedy and Action. Looking at the error bar chart it shows evidence of a genre effect based on the different average ratings per genre from Comedy with the lowest average rating up to Drama/War with the highest average rating. Looking at the data over time it appears that the trend of average ratings is drifting slightly. There may be some evidence of a time effect, but it is not particularly strong.

Results

section that presents the modeling results and discusses the model performance

Conclusion

The objective of the data science Capstone project was to create a movie recommendations system using the MovieLens dataset. After the movielens dataset is split into a test set and a validation set a closer look is taken on the test set data. The exploratory data analysis indicates that movie, user and potentially genre effects are the most promising effects in accurately predicting movie ratings. Plotting ratings against the user and movie data it becomes apparent that the numbers of ratings is not evenly spread across all movies and users. Consequently three different regression models are evaluated based on their RMSE (root mean squared error) as a recommendation system vs. just using the average movie rating as a predictor. The first model uses movie and user effects to predict movie ratings, the second model adds a genre effect to the two previously mentioned factors. The third penalized least squares model accounts for the concentration of ratings across movies and users (or rather the uncertainty which comes from less rated movies or less active users) through regularization. Evaluating the performance of the three models the RMSE improved from 1.060 to 0.8648 showing that using linear regression models as recommendation systems improves performance by around 18.4% (compared to the baseline), with the regularized linear regression model having the best performance. The analysis of this report can be extended to more sophisticated prediction models in future work to evaluated their performance improvement potential.