Spring 2022 B9653 MS Machine Learning Research Project

A Model-Based Recommender System on Amazon Fine Dine Food Dataset

Yue Song | UNI: ys3488 Email: ys3488@columbia.edu

1. Introduction

As an irreplaceable component of the modern business fields, e-commerce now has gained more importance through customers' growing preferences of convenient buying and selling, and the emergence of abundant merchants, including a larger portion of small business owners, who stand out by online transaction only. E-commerce encompasses a wide variety of data, systems, and tools for online buyers and sellers, including mobile shopping and online payment encryption. With the promotion of big data, e-commerce businesses are able to collect and build robust databases on transaction, inventory, shipping, and other related information generated during activities. The implementation of information technology makes it easier for businesses to track and predict their sales growth and therefore adjust current pricing strategies. In addition, some e-commerce tycoons such as Amazon and Walmart gain evolving benefits from the management of customer-related records and feedbacks from large-scale transactions. Customers of Amazon, for example, will always trust and use reviews and ratings from existing purchases to judge their own preferences. In this report, we will explore a recommender system for some Amazon Fine Dine Food Reviews based on the recommendation method of model-based collaborative filter.

The Amazon Fine Dine Food Reviews Dataset[1] consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plaintext review. *Table 1* below gives a brief view of variables in the data.

Variable Name	Data Type	Description
Id	Integer	Raw Id
ProductId	Character	Unique identifier for the product
UserId	Character	Unique identifier for the user
ProfileName	Character	Profile name of the user
HelpfulnessNumerator	Integer {0, 1}	Number of users who found the review helpful
HelpfulnessDenominator	Integer {0, 1}	Number of users who indicated whether they found the review helpful or not
Score	Integer 1~5	Rating between 1 and 5
Time	Integer	Timestamp for the review
Summary	Character	Brief summary of the review
Text	Character	Text of the review

Table 1. Data Type & Description

In this report, we are using R Studio version of 2021.09.0 and R version of 4.1.1 to do an exploratory data analysis and build a recommender system for Amazon Fine Dine Food Reviews Dataset. Since we are planning to build a model-based recommender system, we will include only three variables for our future analysis: "ProductId", "UserId", and "Score". The initial number of observations is 568454, which is quite large may require further examination.

By grouping all the scores by users and products, we find that users will give multiple ratings to the same product and the extreme case is that a frequent buyer gives each product the same number of ratings, up to 11 times. This signals a variation to our model because some users may rate products each time with a similar rating, but some others may rate the same product differently based on experiences. In order to avoid this

[1] The dataset is from https://snap.stanford.edu/data/web-FineFoods.html

variation, it may be helpful to average the scores that each user gives to each product. After the transformation, there are 7650 observations fewer in total, which are repetitive ratings by users on same products. And we are getting 256059 unique users and 74258 unique products after all preprocessing work on the raw data set.

2. Exploratory Data Analysis

In the exploratory data analysis (EDA), we are going to examine the distribution of ratings from different aspects to give a general idea of model we will create later.

First, we are interested in the shape of rating distribution in general. Where is the distribution skewed to? Are there more high ratings or low ratings? Can we get a picture of customer rating behavior? In *Figure 1*, we observe a general left-skewed pattern among all the ratings.

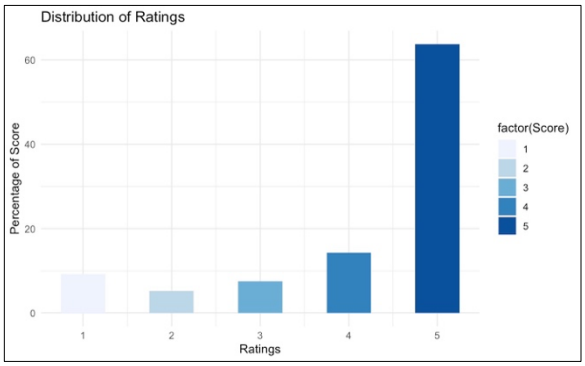

Figure 1. Distribution of Ratings

There's around 62% of ratings that are five-star, and 14% are four-star, 8% are three-star, 5% are two-star, and finally, 10% are one-star. This pattern makes much sense in the real world, because a large group of customers tend to rate consistently high due to convenience if the overall experience is somehow ok, another group of customers then like to take time reflect on their actual experience and may give 3–5-star scores to the products. And the rest of the small group may happen to experience badly during either shipping, consuming, or using the products. They will give one-star scores to some products that others may rate high.

The average rating if we group by users is about 4.17, shown at the blue line in Figure 2.

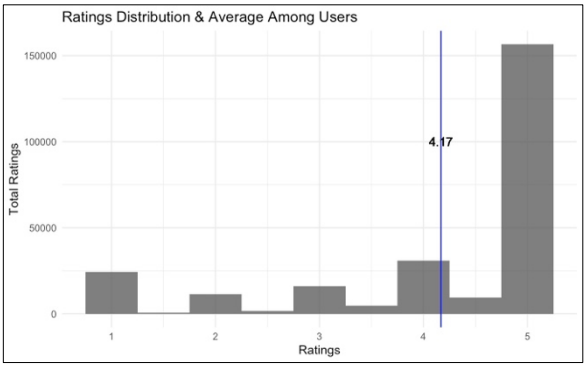

Figure 1. Ratings Distribution & Average Given by Users

Similarly, if we plot the rating distribution and the average by products in Figure 3, we will find the overall shape is also skewed to the left with approximately the same average 4.17. This phenomenon shows that users are not rating the products randomly, which is a good sign to our model because we don't want nonsense rating scores to disturb our model performance.

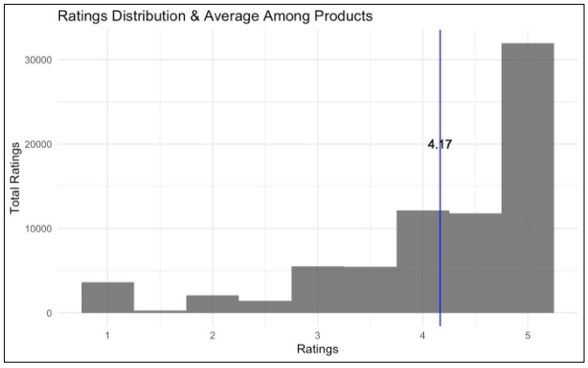

Figure 2. Ratings Distribution & Average Among Products

We are also interested in the rating count distributions and their averages. If we plot the number of ratings per user in Figure 4, we will get an oppositely right-skewed shape, meaning that a large portion of users rate fewer.

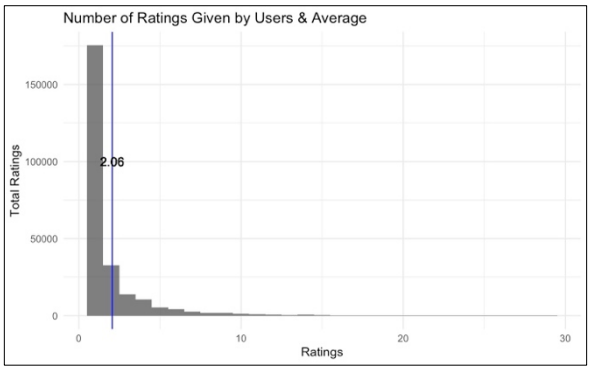

Figure 3. Number of Ratings Given by Users & Average

Note that the average number of ratings given by users is only 2.17. Despite the fun fact above that a user gives all 11 ratings to each of the products he or she purchases, we will find the average number of 2 makes sense in the real world. Very few users will rate the products more than ten times. If the overall buying experience is smooth, most users are just using the products themselves but will not care about feedbacks that they can provide back to other customers.

However, the average number of ratings given to products is high, up to 4.49, shown in *Figure 5*.

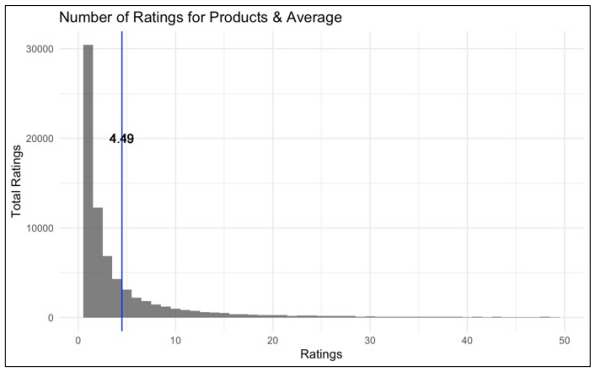

Figure 5. Number of Ratings for Products & Average

This signals that the products themselves may weigh more than user behavior. By confirming this, we may expect a better performance of recommender system that we are going to implement next.

3. Recommender System

We now have a good understanding of our dataset, and the recommender system of model-based collaborative filtering method is implemented next. The model here is called latent factor models[2]. The core is to reduce dimension that can be used directly to predict ratings on their own. The intuition is to exploit dependencies between rows/columns of the ratings matrices to approximate the matrix using a lower dimensional representation that provides satisfactory estimation of the missing entries and occasionally intuitive understanding of the factors driving the data. Common

[2] The latent factor model is discussed in session 5 of MS Machine Learning in Columbia University.

dimensionality reduction techniques are Principal Components Analysis (PCA) and Singular Value Decomposing (SVD). The common idea is to find a set of dominant latent vectors that describe the data in a satisfactory way, and which can be used to estimate missing entries. In this report, we will only focus on the performance of SVD technique.

We will skip the algorithm behind SVD right now and see how the SVD model can produce on this dataset.

First, we transform the preprocessed data frame into a "realRatingMatrix" built in R Studio. The rating matrix is very large at a size of 30.8 Megabyte, so we will choose a portion of ratings to limit the size of our evaluation scheme next for faster execution. According to the rating count distributions in EDA, we decide to only include users that have rated over 30 times and products that are rated more than 50 times.

Second, we implement a cross validation of 5 folds to build the evaluation scheme on the processed rating matrix using a function called "evaluationScheme" from the R package "recommenderlab". The evaluation scheme has the training sets, validation sets ("known" ratings), and test sets ("unknown" or "missing" ratings). We specify the evaluation model to be "SVD" and build the recommender system using the training sets.

Third, we predict using the validation sets and specify 10 items to recommend as the simulation. The validated prediction and the test sets will then be used to calculate the accuracy scores. Table 2 below shows the prediction score on each user, including "RMSE", "MSE", and "MAE". Based on the head only, we will capture three of these users report high scores. If we look back to the raw dataset, we will find that these three users usually give fluctuating scores to even the same products. But the other users with low scores have consistent either high or average scores. This may explain the difference of scores among users. Overall, the recommender system using "SVD" technique performs pretty good on Amazon Fine Dine Food Reviews Dataset.

	RMSE	MSE	MAE
A106ZCP7RSXMRU	0.1995550	0.03982218	0.1995550
A11UPNFYDICF3C	0.3647095	0.13301302	0.2228859
A12ENBT314RFXR	1.1295449	1.27587159	1.0202364
A132ETQPMHQ585	1.7276858	2.98489828	1.6006625
A13MKSASQ6YWL7	1.0073133	1.01468000	0.9122503
A13WOT3RSXKRD5	0.6624678	0.43886360	0.3840699

Table 2. Head Prediction Accuracy for Each User

RMSE	MSE	MAE
0.9407123	0.8849397	0.6847079

Table 3. Average Prediction Accuracy among Users

4. Conclusion

In this report, we introduce the Amazon Fine Dine Food Reviews Dataset and its meaning to the business field. We have done an exploratory data analysis on the dataset and preprocessed it to be used in the recommendation model. We focus on "SVD" technique and the model-based collaborative filtering method on building the system. The model reports well with a good score performance. In further work, we are interested in building different systems such as those using item-based collaborative filtering (IBCF) methods and trying other resampling method on creating the evaluation sets. We also expect to see the effects of customer reviews on product ratings over time. This may involve extracting features and building a content-based recommender system.

Appendix

```
```{r}
#import libraries required
library(dplyr)
library(ggplot2)
library(recommenderlab)

#import the data set and check the top 10 rows
raw <- read.csv("Reviews.csv", header = TRUE)
head(raw, 10)

#check the summary of the raw data set
str(raw)

#create a new dataframe with only UserId, ProductId, and Score that we
will use
df <- raw[, c("UserId", "ProductId", "Score")]
#check the dimension of the dataframe
dim(df)

#find out how many ratings a single user can have for one product
df %>%
  group_by(UserId, ProductId) %>%
  summarise(numberOfRatings = n()) %>%
  arrange(desc(numberOfRatings)) %>%
  head(20)


#create a new data frame using the rounded mean score for each user-
product pair as the rating
df_1 <- df %>%
          group_by(UserId, ProductId) %>%
          summarise(Score = round(mean(Score)))

head(df_1, 10)
#check out the reduction in the number of user-product pair
dim(df_1)

#check out the number of unique users
df_1$UserId %>% unique() %>% length
print("There are 256059 unique users")
#check out the number of unique products
df_1$ProductId %>% unique() %>% length
print("There are 74258 unique products.")

#create a bar plot to see the percentage distribution of ratings
ggplot(data = df_1, aes(x = Score, y = (..count..)/sum(..count..)*100,
fill = factor(Score))) +
  geom_bar(width = 0.5) +
  theme_minimal() +
  scale_fill_brewer(palette = "Blues") +
  ggtitle("Distribution of Ratings") +
  labs(y = "Percentage of Score", x = "Ratings")
```
```

```
#create a histogram to show the distribution of ratings given by users
and the average
df_1 %>%
  group_by(UserId) %>%
  summarise(meanScore = mean(Score)) %>%
  ggplot() +
    geom_histogram(aes(x = meanScore), binwidth = .5, alpha = .8,
position = "identity") +
    geom_vline(aes(xintercept = mean(meanScore)), color = "blue") +
    geom_text(aes(x = mean(meanScore), label = round(mean(meanScore), 2),
y = 100000)) +
    theme_minimal() +
    ggtitle("Ratings Distribution & Average Given by Users") +
    labs(y = "Total Ratings", x = "Ratings")

#create a histogram to show the distribution of ratings of products and
the average
df_1 %>%
  group_by(ProductId) %>%
  summarise(meanScore = mean(Score)) %>%
  ggplot() +
    geom_histogram(aes(x = meanScore), binwidth = .5, alpha = .8,
position = "identity") +
    geom_vline(aes(xintercept = mean(meanScore)), color = "blue") +
    geom_text(aes(x = mean(meanScore), label = round(mean(meanScore), 2),
y = 20000)) +
    theme_minimal() +
    ggtitle("Ratings Distribution & Average Among Products") +
    labs(y = "Total Ratings", x = "Ratings")

#create a histogram to show the distribution of the number of ratings
given by users and the average


df_1 %>%
  group_by(UserId) %>%
  summarise(scoreCount = n()) %>%
  filter(scoreCount < 30) %>%
  ggplot() +
    geom_histogram(aes(x = scoreCount), binwidth = 1, alpha = .8,
position = "identity") +
    geom_vline(aes(xintercept = mean(scoreCount)), color = "blue") +
    geom_text(aes(x = mean(scoreCount), label = round(mean(scoreCount),
2), y = 100000)) +
    theme_minimal() +
    ggtitle("Number of Ratings Given by Users & Average") +
    labs(y = "Total Ratings", x = "Ratings")

#create a histogram to show the distribution of the number of ratings for
products and the average
df_1 %>%
  group_by(ProductId) %>%
  summarise(scoreCount = n()) %>%
  filter(scoreCount < 50) %>%
  ggplot() +
    geom_histogram(aes(x = scoreCount), binwidth = 1, alpha = .8,
position = "identity") +
    geom_vline(aes(xintercept = mean(scoreCount)), color = "blue") +
    geom_text(aes(x = mean(scoreCount), label = round(mean(scoreCount),
2), y = 20000)) +
    theme_minimal() +
    ggtitle("Number of Ratings for Products & Average") +
    labs(y = "Total Ratings", x = "Ratings")
```

```
#transform the processed data set into a rating matrix
rating_matrix <- as(as.data.frame(df_1), "realRatingMatrix")
#choose a portion of ratings to limit the size of our evaluation scheme
for faster execution
rating_matrix <- rating_matrix[rowCounts(rating_matrix) > 30,
colCounts(rating_matrix) > 50]
rating_matrix <- rating_matrix[rowCounts(rating_matrix) > 10,]
#create the evaluation scheme using cross validation method
eval_sets <- evaluationScheme(rating_matrix, method = "cross-validation",
k = 5, given = 10)

set.seed(1)
model_to_evaluate <- "SVD"
model_parameters <- NULL
eval_recommender <- Recommender(data = getData(eval_sets, "train"),
                                method = model_to_evaluate,
                                parameter = model_parameters)
items_to_recommend <- 10
eval_prediction <- predict(object = eval_recommender,
                           newdata = getData(eval_sets, "known"),
                           n = items_to_recommend,
                           type = "ratings")

#calculate the prediction accuracy by users
eval_accuracy <- calcPredictionAccuracy(
  x = eval_prediction,
  data = getData(eval_sets, "unknown"),
  byUser = TRUE)


head(eval_accuracy)
#calculate the average prediction accuracy (not by users)
eval_accuracy <- calcPredictionAccuracy(
  x = eval_prediction,
  data = getData(eval_sets, "unknown"),
  byUser = FALSE)
eval_accuracy
```