Predicting Food Preferences Using Recommendation Systems: A Model Comparison Approach

M. Sääksjärvi

1/22/2019

Executive Summary

Recommender systems are systems that help users find new products and services based on their prior preferences. They are widely used in areas such as movies and books, and have proven successful in their ability to drive future sales. Lesser is known about the performance of recommendation systems in other areas, e.g., in the context of food. As compared with preferences for durable goods or entertainment items, preferences for food items are established early in life. Predicting preference for food items based on recent purchases might therefore be relatively challenging. Using a dataset consisting of 568,454 food reviews Amazon, I build a recommendation system for food preferences. Using matrix factorization and regularization, I estimate five difference models for this purpose, which are compared to each other on the basis of RMSE. The results show that the best performing model is a matrix factorization model that includes user and product bias.

Introduction

Recommender systems (RS) are used to help users find new items or services (e.g., books, music, transportation) based on information about the user and his/her past behavior (Adomavicius and Tuzhilin 2005). Essentially, recommender systems apply statistical techniques

to make predictions about the users future behavior on the basis on previously recorded data (Sarwar, Karyptis, Konstan, and Riedl 2000). Recommender systems have become popular in recent years. Several market industry leaders use recommender systems (e.g., Amazon, Netflix, and Pandora), as such systems are good for business: they help improve conversion rates, promote cross-selling, and foster the relationship with the user (Shafer, Konstan, and Riedl 2001). Recommender systems involve machine learning, "the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to 'learn'." (Kohavi and Provost 1998). Machine learning centers on the creation of algorithms that can make predictions based on a learning process (Mitchell 1999), and can thereby be used for predicting a user's future preferences. Recommendation systems have been widely tested in areas such as movies and books (e.g., the Netflix challenge, Lohr 2009). A lesser known context for recommendation systems is the food market. The food market is a lucrative business: revenues in the food and beverages segment amounts to US$18,968m in 2019, and is expected to show an annual growth rate of 10.0%, resulting in a market volume of US$27,760m by 2023 (Statista 2019). Consumer preferences for food are established relatively early in life relative to other preferences (e.g., for books and movies) (Gugusheff, Ong, and Muhlhausler 2015), which may make them difficult to influence or predict. Food preferences are also interesting to try to predict as food represents the struggle between wants and needs: "wants" such as junk food and chocolate vs. "needs" such as healthy foods and vegetables. This could make preferences fluctuate widely depending on what goals consumers' currently hold active.

The Data

For analyzing food preferences, a dataset was sourced from Kaggle (https://www.kaggle.com/snap/amazon-fine-food-reviews). The datasetcontains 568,454 food reviews collected from Amazon. The data contains the following fields:

- Id: The row ID in the file
- ProductId: An ID assigned to the product
- UserId: An ID assigned to the unique user
- ProfileName: The profile name of the user
- HelpfulnessNumerator: The number of users who found the review helpful
- HelpfulnessDenominator: The number of users who indicated whether they found the review helpful or not
- Score: Rating given to the product between 1 and 5
- Time: Timestamp for the review
- Summary: Brief summary of the review
- Text: Text of the review

The Goal

The goal is to predict food preferences by training a machine learning algorithm capable of predicting such preferences. To reach this goal, the data is split into two sets, a training set, and a test set using a 90-10 split, with 90% of the data being allocated to training (dataset r_train), and 10% to testing (dataset Reviewtest). The aim is to reach an RMSE (root mean square error) that is as low as possible when predicting the preferences through the use of various libraries in R and various statistical techniques.

Methods and Analysis

Data Preparation

```
#Loading libraries
library(readr)
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyv
erse 1.2.1 ──
```

```
## ✓ ggplot2 3.1.0     ✓ purrr   0.2.5
## ✓ tibble  1.4.2     ✓ dplyr   0.7.8
## ✓ tidyr   0.8.1     ✓ stringr 1.3.1
## ✓ ggplot2 3.1.0     ✓ forcats 0.3.0

## ── Conflicts ─────────────────────────────────────────────────────────────────────────────── tidyverse_
conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

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

library(tidyr)
library(ggplot2)
```

Getting Data

```
#Getting the data
Reviews <- read_csv("~/Desktop/Reviews.csv")

## Parsed with column specification:
## cols(
##   Id = col_integer(),
##   ProductId = col_character(),
##   UserId = col_character(),
##   ProfileName = col_character(),
##   HelpfulnessNumerator = col_integer(),
##   HelpfulnessDenominator = col_integer(),
##   Score = col_integer(),
##   Time = col_integer(),
##   Summary = col_character(),
##   Text = col_character()
## )

#Define scores and ProductId as numeric
Reviews$Score <- as.numeric(Reviews$Score)
Reviews$ProductId <- as.factor(Reviews$ProductId)
Reviews$ProductId <- as.numeric(Reviews$ProductId)

#Create train and test sets, with 10% of the data allocated to the test set
set.seed(1)
test_index <- createDataPartition(y = Reviews$Score, times = 1, p = 0.1, list = FALSE)
r_train <- Reviews[-test_index,]
r_test <- Reviews[test_index,]
```

```
#Make sure ProductId and UserId in the test set are also in the train set
Reviewtest <- r_test %>%
  semi_join(r_train, by = "ProductId") %>%
  semi_join(r_train, by = "UserId")

#Add rows from removed test set back into the train set
removed <- anti_join(r_test, Reviewtest)

## Joining, by = c("Id", "ProductId", "UserId", "ProfileName", "HelpfulnessNumerator", "HelpfulnessDe
nominator", "Score", "Time", "Summary", "Text")

r_train <- rbind(r_train, removed)

#Remove files that are no longer needed
rm(Reviews, test_index, r_test, removed)
```

Exploratory Data Analysis

We start by examining the structure and dimensions of the training set, and the number of users and products included in it.

```
#Examine the structure of the training set
head(r_train)

## # A tibble: 6 x 10
##      Id ProductId UserId ProfileName HelpfulnessNume… HelpfulnessDeno…
##   <int>     <dbl> <chr>  <chr>                 <int>            <int>
## 1     1     27620 A3SGX… delmartian                 1                1
## 2     2     72384 A1D87… dll pa                     0                0
## 3     3     15268 ABXLM… "Natalia C…                1                1
## 4     4     19719 A395B… Karl                       3                3
## 5     5     69008 A1UQR… "Michael D…                0                0
## 6     6     69008 ADT0S… Twoapennyt…                0                0
## # ... with 4 more variables: Score <dbl>, Time <int>, Summary <chr>,
## #   Text <chr>

#Examine the dimensions of the dataset
dim(r_train)

## [1] 531401     10

summary(r_train)

##        Id          ProductId       UserId         ProfileName
##  Min.   :     1   Min.   :    1   Length:531401      Length:531401
##  1st Qu.:142121   1st Qu.:15909   Class :character   Class :character
##  Median :284064   Median :32806   Mode  :character   Mode  :character
##  Mean   :284205   Mean   :34687
```

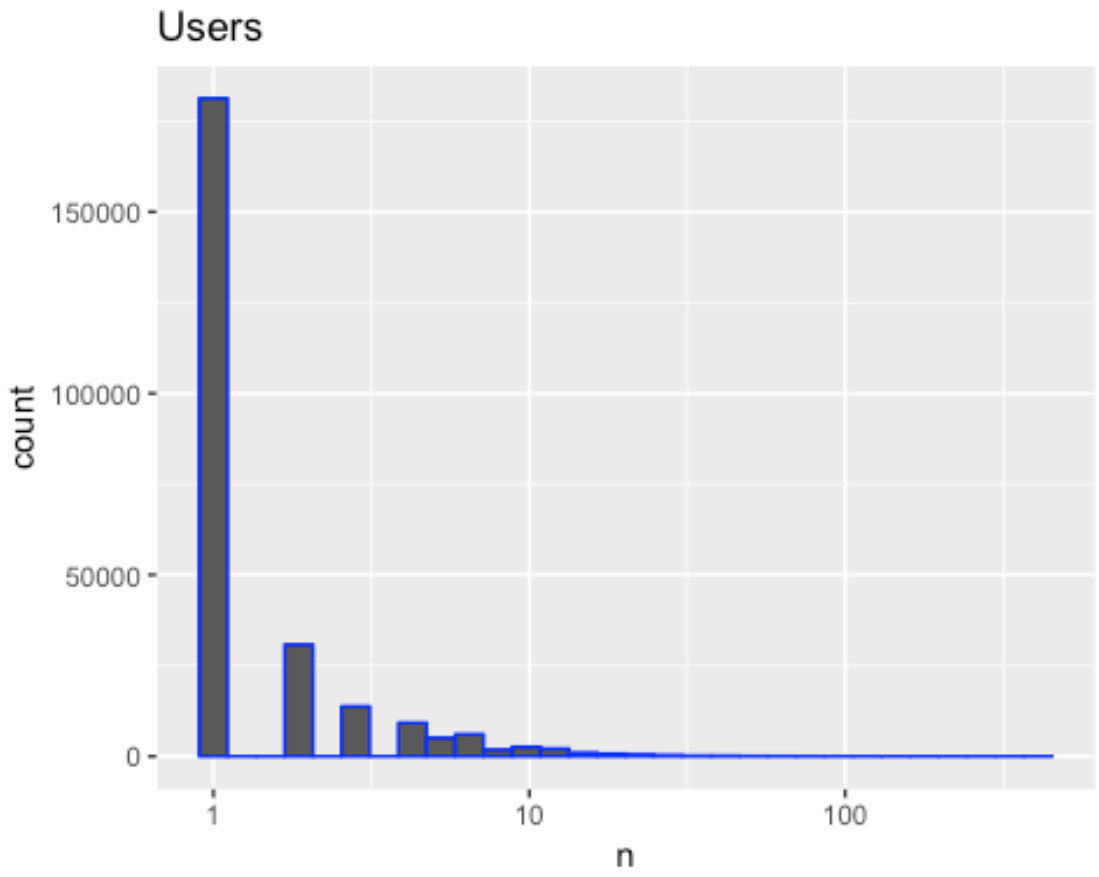
```
##  3rd Qu.:426443   3rd Qu.:51295
##  Max.   :568454   Max.   :74258
##  HelpfulnessNumerator HelpfulnessDenominator     Score
##  Min.   :  0.00       Min.   :  0.000        Min.   :1.000
##  1st Qu.:  0.00       1st Qu.:  0.000        1st Qu.:4.000
##  Median :  0.00       Median :  1.000        Median :5.000
##  Mean   :  1.74       Mean   :  2.225        Mean   :4.183
##  3rd Qu.:  2.00       3rd Qu.:  2.000        3rd Qu.:5.000
##  Max.   :866.00       Max.   :923.000        Max.   :5.000
##       Time             Summary              Text
##  Min.   :9.393e+08   Length:531401      Length:531401
##  1st Qu.:1.271e+09   Class :character   Class :character
##  Median :1.311e+09   Mode  :character   Mode  :character
##  Mean   :1.296e+09
##  3rd Qu.:1.333e+09
##  Max.   :1.351e+09
```

```
#Numbers of users and products
r_train %>%
  summarize(n_users = n_distinct(UserId),
            n_products = n_distinct(ProductId))
```

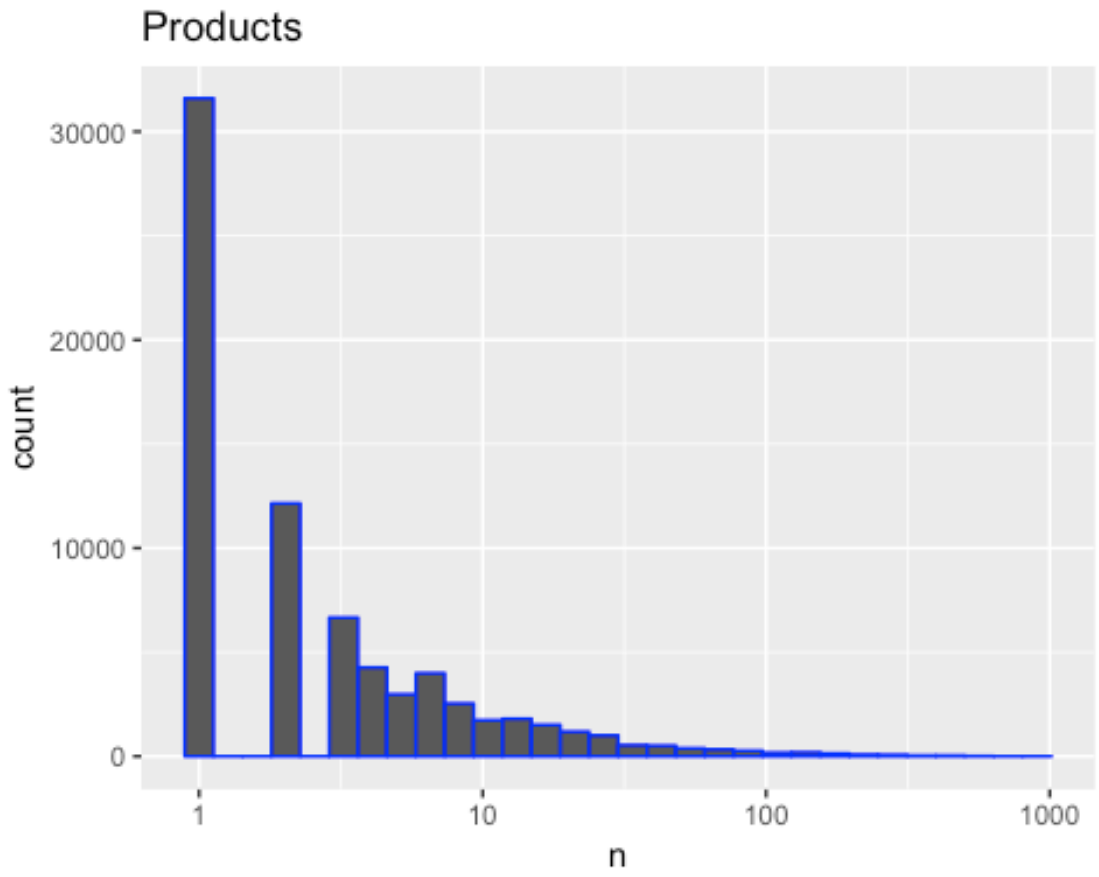
```
## # A tibble: 1 x 2
##   n_users n_products
##     <int>      <int>
## 1  256059      74258
```

The plots below show the distribution of users, products, and ratings. As can be seen in the plots below, most users have rated a single product, and most producst have been rated only once.

```
#Distribution of users
r_train %>%
  count(UserId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  ggtitle("Users")
```

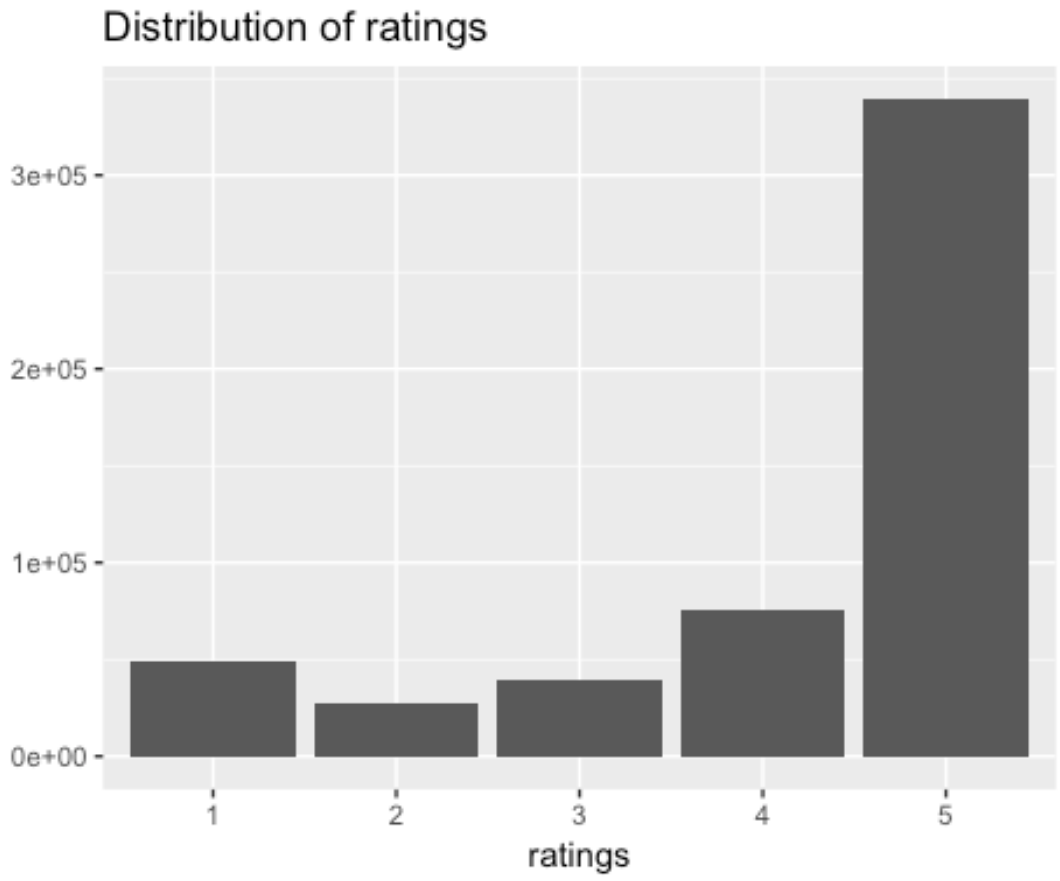



```r
#Distribution of products
r_train %>%
  count(ProductId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  ggtitle("Products")
```



The distribution of rating shows that most users gave positive scores to the products they rated, which can also be seen in the positively skewed mean (4.18 on a scale from 1-5). The top rated product is Quaker's Oatmeal cookies (product 71171).

```
#Distribution of ratings
ratings <- as.vector(r_train$Score)
ratings <- factor(ratings)
qplot(ratings) +
  ggtitle("Distribution of ratings")
```




```
#Mean and sd of review scores
mean(r_train$Score)

## [1] 4.182548

sd(r_train$Score)

## [1] 1.312661

#Top 10 rated products
r_train %>% group_by(ProductId) %>%
  summarize(n = n()) %>%
  arrange(desc(n))

## # A tibble: 74,258 x 2
##    ProductId     n
##        <dbl> <int>
##  1    71171   903
##  2    46206   600
##  3    42258   584
##  4    42265   584
##  5    42264   579
##  6    37899   573
##  7    28625   546
##  8    16946   520
```

```
##  9    35244   518
## 10    23310   517
## # ... with 74,248 more rows
```

Model Comparison

Predicting results based on the average rating

Using a matrix factorization approach, we start by building the simplest possible model for a recommendation system. The simplest possible model is the average rating: it assumes the same rating for all products, regardless of the user or the product. Any subsequent model should be able to to improve the RMSE value of this simplest possible model. As such, this model will serve as a baseline for our RMSE comparison.

```
#RMSE metric
RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

#Predicting results from average rating
mu <- mean(r_train$Score)
model_1_rmse <- RMSE(Reviewtest$Score, mu)
rmse_results <- data_frame(method = "Just the average", RMSE = model_1_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.278106

The RMSE value for this model is relative high, 1.2781064.

Including product bias into the model

To minimize this value, we continue by including the product bias into the model.

```
#Including product bias
mu <- mean(r_train$Score)
product_avgs <- r_train %>%
  group_by(ProductId) %>%
  summarize(b_i = mean(Score- mu))
```

```
predicted_ratings <- mu + Reviewtest %>%
  left_join(product_avgs, by="ProductId") %>%
  .$b_i

#Estimating RMSE for product bias
model_2_rmse <-RMSE(predicted_ratings, Reviewtest$Score)
rmse_results <- bind_rows(rmse_results, data_frame(method= "Product Effect Model", RMSE = model
_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.278106
Product Effect Model	1.280495

The value of RMSE did not change much, it actually slightly decreased.

Including user bias into the model

We continue by including the user bias.

```
#Including user and product bias
user_avgs <- r_train %>%
  left_join(product_avgs, by="ProductId") %>%
  group_by(UserId) %>%
  summarize(b_u = mean(Score - mu - b_i))

predicted_ratings <- Reviewtest %>%
  left_join(product_avgs, by="ProductId") %>%
  left_join(user_avgs, by="UserId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

#Estimating RMSE for user and product bias
model_3_rmse <- RMSE(predicted_ratings, Reviewtest$Score)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Product + User Effects Model", RMSE
= model_3_rmse))

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

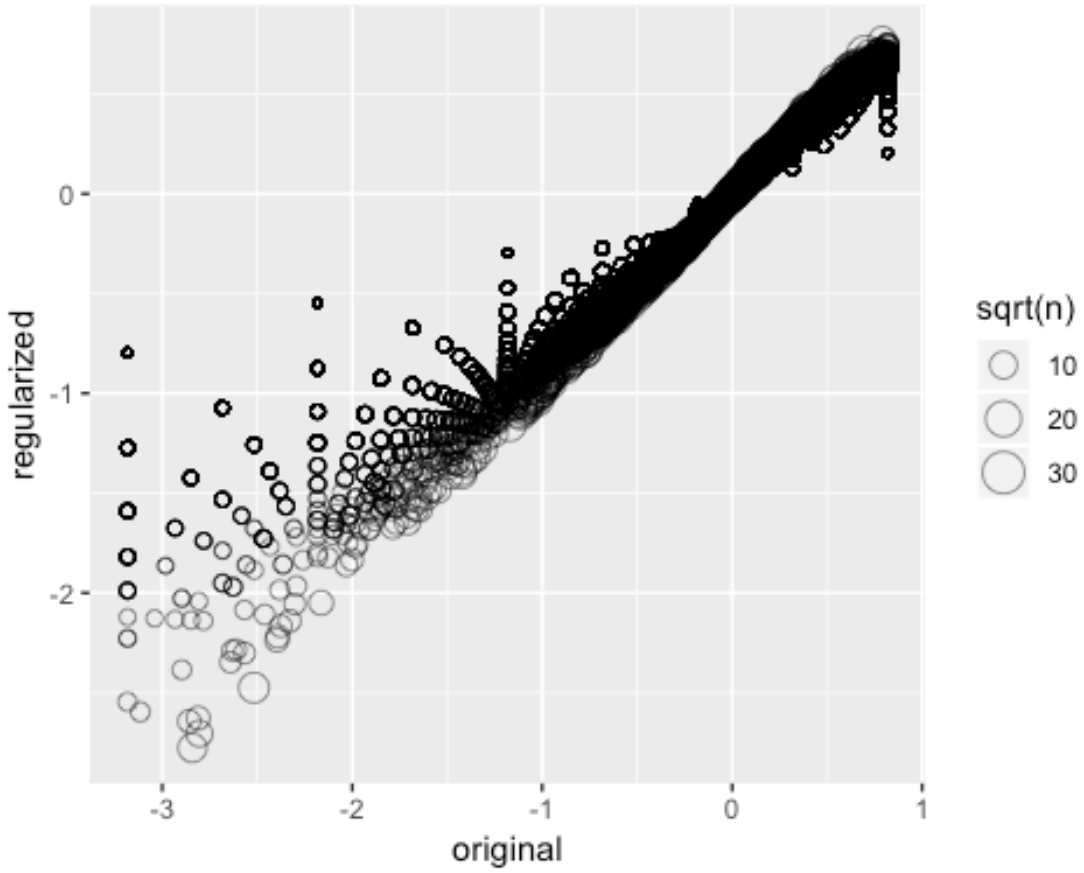
method	RMSE
Just the average	1.2781064
Product Effect Model	1.2804946
Product + User Effects Model	0.9411876

Now we see a significant improvement in our RMSE: it decreased to 0.9411876.

Regularization

We continue our optimization efforts by switching over to regularization. We estimate the regularization model using a lambda of 3.

```
#Switching to regularization, estimating lambda
lambda <- 3
mu <- mean(r_train$Score)
food_reg_avgs <- r_train %>%
  group_by(ProductId) %>%
  summarize(b_i = sum(Score - mu)/(n()+lambda), n_i = n())

#Plot of regularized estimates vs. least square estimates
data_frame(original = product_avgs$b_i,
           regularized = food_reg_avgs$b_i,
           n = food_reg_avgs$n_i) %>%
  ggplot(aes(original, regularized, size =sqrt(n))) +
  geom_point(shape = 1, alpha = 0.5, color = "black")
```

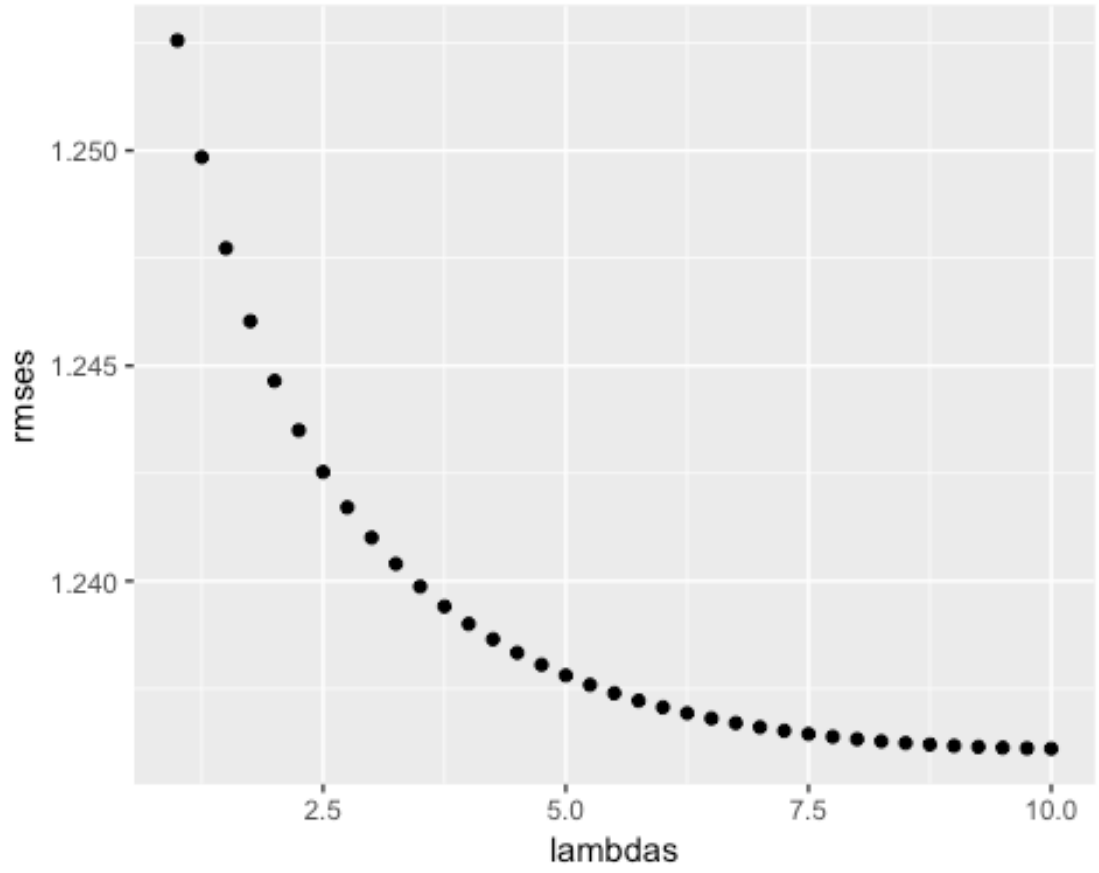



```
predicted_ratings <- Reviewtest %>%
  left_join(food_reg_avgs, by="ProductId") %>%
  mutate(pred = mu + b_i) %>%
  .$pred

#Estimating RMSE for regularization
model_4_rmse <- RMSE(predicted_ratings, Reviewtest$Score)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularization", RMSE = model_4_rm
se))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.2781064
Product Effect Model	1.2804946
Product + User Effects Model	0.9411876
Regularization	1.2410069

Our results did not improve. So we continue by trying to find the optimal lambda to use for our predictions via cross-fertilization.

```
#Choosing the optimal lambda via cross-fertilization
lambdas <- seq(1, 10, 0.25)
mu<- mean(r_train$Score)
just_the_sum <- r_train %>%
  group_by(ProductId) %>%
  summarize(s = sum(Score-mu), n_i = n())

rmses <- sapply(lambdas, function(l) {
  predicted_ratings <- Reviewtest %>%
    left_join(just_the_sum, by="ProductId") %>%
    mutate(b_i=s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, Reviewtest$Score))
})
#Plotting lambdas
qplot(lambdas, rmses)
```




```
#Optimal lambda
l <- lambdas[which.min(rmses)]
l

## [1] 10
```

Regularization + model

Now that we have the optimal lambda, we can include it in our prediction. We estimate a new model based on prodcut and user effects and the optimal lambda we calculated.

```
#Including product and user effects into the regularized model (Regularization+)
 b_i <- r_train %>%
   group_by(ProductId) %>%
   summarize(b_i = sum(Score - mu)/(n()+l))

 b_u <- r_train %>%
   left_join(b_i, by="ProductId") %>%
   group_by(UserId) %>%
   summarize(b_u = sum(Score - b_i- mu)/(n()+l))

 predicted_ratings <- Reviewtest %>%
   left_join(b_i, by="ProductId") %>%
   left_join(b_u, by="UserId") %>%
   mutate(pred = mu + b_i + b_u) %>%
   .$pred

#Estimating RMSE for regularization+ model
model_5_rmse <- RMSE(predicted_ratings, Reviewtest$Score)
rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularization+", RMSE = model_5_r
mse))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.2781064
Product Effect Model	1.2804946
Product + User Effects Model	0.9411876
Regularization	1.2410069
Regularization+	1.0351550

Our results with optimal lambda are slightly improved as compared with our basic regularization model, but they still fall short of the matrix factorization model with product and user effects.

Final Model

Based on a comparison of five models – Just the average, Product effects model, User effects model, Regularization and Regularization+ – we may conclude that a matrix factorization model with product and user effects (User effect model) is the best performing model.

Conclusions

In this report, we focused on creating a recommendation system based on machine learning algorithms in the context of food recommendations. Relative to other areas (e.g., movie recommendations), food recommendations have received relatively little attention. We estimated food recommendations using a dataset of 568,454 food reviews Amazon. Exploratory data analysis revealed a highly idiosyncratic dataset, as most food items had only been rated once, with most users having only rated one product. This is in stark constrast to movie recommendation systems, in which heavy users (movie fanatisc) tend to rate a relatively larger proportion of the products(movies) in a dataset, and represents a limitation of the current report. For estimating food recommendations, we compared five different models usig matrix factorization and regularization. We found that a matrix factorization model that accounts for product and user bias outperforms the other matrix factorization models. Models estimated on the basis of regularization also did not improve on the matrix factorization model with product and user bias.

References

Adomavicius, G., and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

Gugusheff, J. R., Ong, Z. Y., & Muhlhausler, B. S. (2015). The early origins of food preferences: targeting the critical windows of development. The FASEB Journal, 29(2), 365-373.

Kohavi, R. and Provost, F. (1998) Glossary of terms. Machine Learning, 30, 271-274.

Lohr, S. (2009) Netflix awards $1 million prize and starts a new contest. New York Times. Available at https://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-startsa-new-contest/.

Mitchell, T. M. (1999). Machine learning and data mining. Communications of the ACM, 42 (11), 30-36.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In Proceedings of the 2nd ACM conference on Electronic commerce, 158-167.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. Data mining and knowledge discovery, 5(1-2), 115-153.

Statista (2019). Food & beverages outlook. Available at https://www.statista.com/outlook/253/109/food-beverages/united-states