DATA 622: Final Project

Section 2, Group 3

12/10/2021

Introduction

Overview

Through lockdowns and quarantines, the last two years have resulted in many Americans becoming more home-bound than they had been previously and with this change have come a myriad of health concerns. One such concern that has always been in the public eye concerns Americans' typical diet. Our group opted to work with an initially large data set of recipes hosted on Kaggle and originally sourced from Food.com site. By exploring the data, we settled on a specific public health theme, namely, utilizing nutritional data to predict the fiber content of a given recipe. A citation, complete with link, for this data set can be found in our brief references section towards the end of this report.

The initial data set consists of over 500,000 recipes and 28 variables. We would utilize sampling methods to reduce our computational overhead during our modeling, but an interesting and important note about this data set is that there is a massive representation of food recipes and categories present within it. We also utilized a secondary data set of around 1.4 million user reviews, however, we ultimately only used it to assess the feasibility of an initial (and ultimately abandoned) potential trajectory for this project.

Problem Statement/Project Goal

Our goal was to create a model that would help identify recipes that would predict the fiber content a given recipe, thereby empowering content creators and users in generating and consuming healthier recipes. By doing this, we hope to promote public health through the accessible and grassroots act of home cookery. This is not intended as a cure-all, but rather an additional method or tool in the overall pursuit of improving public health.

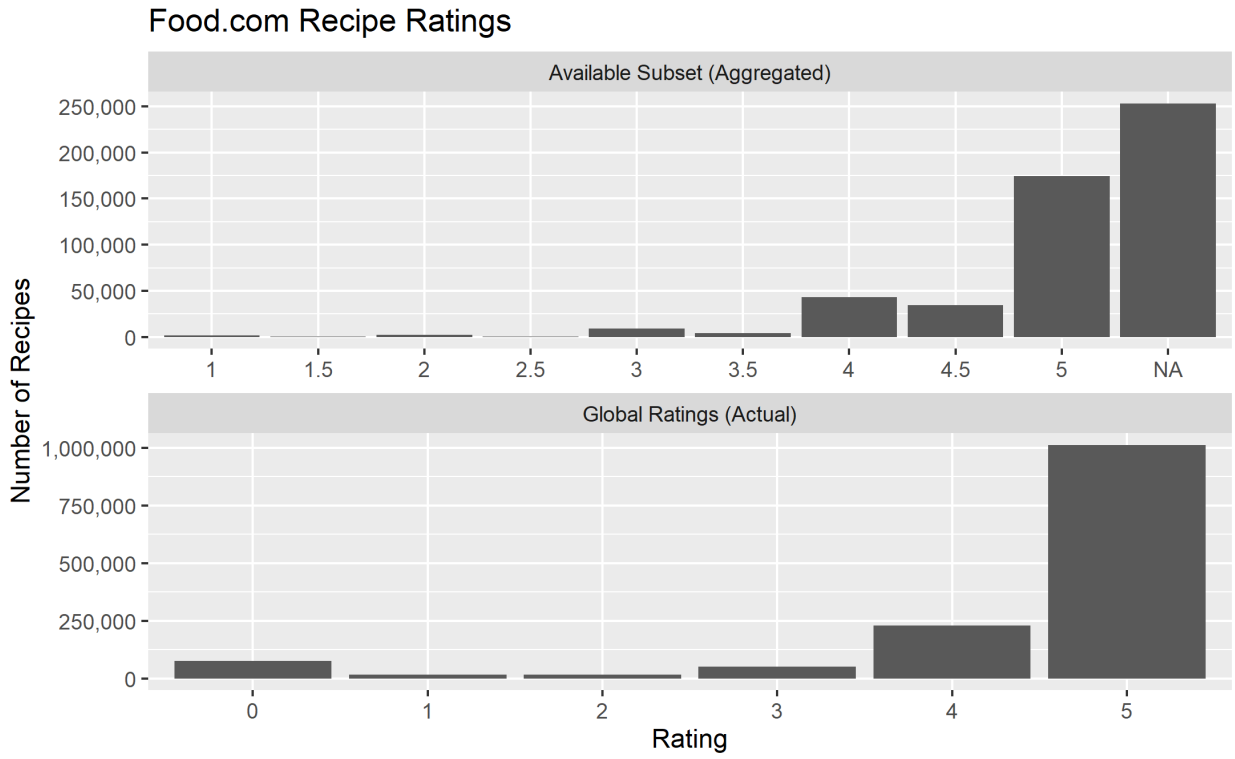
Despite

Exploratory Data Analysis

While strategic sampling is a general best practice, this specific data set was large and we needed to be intentional in our approach to selecting observations. While we would eventually settle on a particular set of predictors and a target variable, we initially had little practical knowledge of the data set. We explored the data to both develop a sense of feasible modeling paths as well as an overall understanding of the data's limitations.

An initial point of interest was aggregated ratings for the recipes. In visualizing ratings (global actual reviews across the site and aggregated figures for the available recipes), there were two primary revelations:

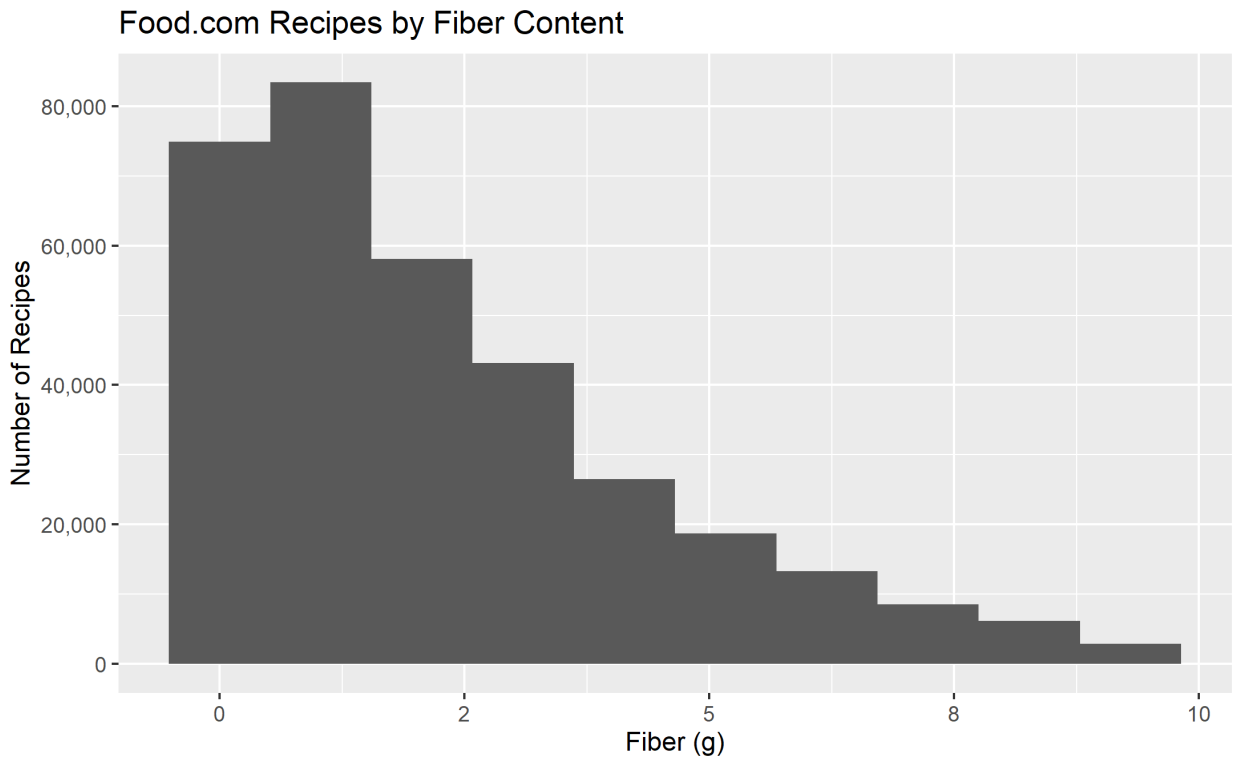
- Users seem to take an all-or-nothing approach to reviewing. That is, 5-star ratings hugely overwhelm all other ratings.
- Nearly half of our given recipe data set is missing an aggregate rating (and are not included in the review data).



While having such a large proportion of missing ratings might be alarming, dropping those observations from use would still net a sizable data set to work with. More troubling is that the ratings themselves appear to follow the afforementioned "all or nothing" approach. That is, users tend only to review what they feel very positively about. Downstream analysis lead to the eventual abandonment of reviews/ratings as an area of interest due to its seeming lack of relationship with rich variety of nutritional variables present in the data coupled with the reality that a binary outcome on recipe success was not particularly interesting.

On the other hand nutritional variables provided quantifiable, numeric measurements for each recipe while many of the non-nutritional variables are identifying elements utilized by the website the data were sourced from. There were a number of text variables also available, but we found the overhead required to mine these variables for useful features to be excessive. We ultimately decided to move forward with our public-health oriented approach and building out a model to predict the fiber content of a recipe based on its other nutritional dimensions.

We can learn about this target variable by visualizing it as we've done below:



The distribution of fiber content across the data set handily illustrates our case in point - the available recipes are overwhelmingly low in fiber. Whatever the reason for this distribution, the practical effect is that even the most diet-conscious home cook looking for high-fiber recipes will have a hard time finding them exploring the site the old fashioned way. According to the USDA's *Dietary Guidelines for Americans, 2020 - 2025*, adults are generally recommended to take in more than 20g of fiber a day. Even if we assume each recipe represented a single meal, the difficulty of identifying higher-fiber meals makes this guideline difficult to achieve by simple recipe browsing alone. Some outside help would be needed to guide users to healthier options.

There is insufficient context for us to impute missing values for any of the nutritional variables we wished to work with, and we had a very large data set to begin with, so rows with missing values were dropped from use. We also opted to ignore publication date, review counts and recipe servings which should all have no impact on the nutritional value of a given recipe. In particular, recipe servings were deemed subjective as these counts rely on too many elements external to the data (e.g. dish size, portion size, etc.). Lastly, outliers were detected and removed by evaluating rows with values outside a range of three mean absolute deviations from the median for each nutritional variable.

Knowing we wished to utilize the nutritional variables, we assessed the correlation between them and our fiber-content target variable. We can see the output from R below:

```
##                     Calories FatContent SaturatedFatContent CholesterolContent
## Calories           1.0000000 0.83160932          0.69880127         0.56339079
## FatContent         0.8316093 1.00000000          0.83833030         0.54668529
## SaturatedFatContent 0.6988013 0.83833030          1.00000000         0.58666647
## CholesterolContent  0.5633908 0.54668529          0.58666647         1.00000000
## SodiumContent       0.5343196 0.43334241          0.36837623         0.37763093
## CarbohydrateContent 0.6542044 0.23535657          0.20321559         0.11044193
## FiberContent        0.4075867 0.20015988          0.08238757        -0.01417622
## SugarContent        0.1297412 0.01014508          0.03955369        -0.04460569
## ProteinContent      0.6947309 0.51626624          0.42090707         0.66905835
```

```
##                       SodiumContent CarbohydrateContent FiberContent SugarContent
## Calories                 0.53431958           0.6542044   0.40758674   0.12974119
## FatContent               0.43334241           0.2353566   0.20015988   0.01014508
## SaturatedFatContent      0.36837623           0.2032156   0.08238757   0.03955369
## CholesterolContent       0.37763093           0.1104419  -0.01417622  -0.04460569
## SodiumContent            1.00000000           0.2923928   0.27440404  -0.09193649
## CarbohydrateContent      0.29239279           1.0000000   0.55236237   0.37987634
## FiberContent             0.27440404           0.5523624   1.00000000   0.08654518
## SugarContent            -0.09193649           0.3798763   0.08654518   1.00000000
## ProteinContent           0.54487259           0.2384846   0.24331764  -0.14755706
##                       ProteinContent
## Calories                   0.6947309
## FatContent                 0.5162662
## SaturatedFatContent        0.4209071
## CholesterolContent         0.6690584
## SodiumContent              0.5448726
## CarbohydrateContent        0.2384846
## FiberContent               0.2433176
## SugarContent              -0.1475571
## ProteinContent             1.0000000
```

Specifying fiber content as our dependent or target variable, we can see that there is a moderate relationship between it and both the caloric and carbohydrate content of a recipe. Fat, sodium and protein content also showed a set of weak relationships with fiber content. It is notable that multicollinearity was present across the various nutritional variables, which makes intuitive sense given the fact that the underlying nutritional values of these recipes are an amalgamation of their ingredients and cooking methods.

Data Preparation

Given the interrelated nature of the nutritional variables, we opted to conduct dimensionality reduction via principal component analysis (PCA) as a first step. This will provide us the benefit of not only side-stepping multicollinearity issues by utilizing an orthogonal transformation. We will also simplify our downstream modeling process as we will work with fewer variables. We lose interpretability by doing this, as the principal components themselves will be difficult to express in words. Tools like biplots might be useful in a diagnostic context, but are rarely of any utility to a non-technical audience.

In our case, however, none of this really matters - we're interested in predictive ability alone and have no real requirement of explanatory ease. The results of our PCA, with scaled and centered values, can be reviewed below.

```
## Importance of components:
##                           PC1    PC2    PC3     PC4    PC5     PC6    PC7
## Standard deviation     2.0207 1.1949 0.9479 0.79370 0.7122 0.54148 0.3805
## Proportion of Variance 0.5104 0.1785 0.1123 0.07874 0.0634 0.03665 0.0181
## Cumulative Proportion  0.5104 0.6889 0.8012 0.87992 0.9433 0.97997 0.9981
##                           PC8
## Standard deviation     0.12414
## Proportion of Variance 0.00193
## Cumulative Proportion  1.00000
```

Based on the summary output above, we can see that half of the variability is explained within the first principal component (PC). Including more of the initial PCs, we can up this proportion and achieve just under 95% with the first five PCs. Regardless of our approach downstream, utilizing up to a handful of these variables means we've minimized our number of predictors by at least half of the original count.

Modeling

Our modeling approach began with the creating both a training and test data set from a randomized sample of a smaller subset of the overall data. This was done to minimize computational overhead as much as to permit us to utilize our given model (discussed in the next paragraph) more readily. In this way, we utilized 1,000 observations split 80% and 20% among the training and test data sets respectively.

Using the `e1071` library, a function was prepared to optimize the hyperparameters of our chosen predictive model - Support Vector Machines (SVM). Through the optimization process, we passed through various potential values for the cost, tolerance, gamma and epsilon hyperparameters, eventually settling on a series of values that can be reviewed in the adjoining code. We also chose to implement 5-fold cross validation to help improve our end performance. We manually experimented by running this process across the different available kernels for this library, and ultimately settled on the radial basis function (RBF) kernel for our final model.

Because this was a regression model, performance was evaluated utilizing the typical suite of metrics - mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) utilizing another custom-written, wrapper function built around the `Metrics` library. The dominant factor in our evaluation was performance with regards to RMSE, largely because the end result is interpreted in the same units as the output itself. That is, we can use RMSE as an anticipated window of how "off" our predictions can be expected to be. Our RMSE was about 1.5 or 1.7 grams of fiber against the training and testing data, respectively.

The drop in RMSE is to be expected, but given the distribution of values we'd seen during EDA, this is an imperfect but still useful degree of error to work with given our use case. Further, this drop is not immense given the units of measure, given the fact that the existing possible range might be between 0-10g of fiber. In a sense, the model's output can be readily utilized as a means of anticipating whether a given recipe will be low, moderate or high fiber with a good degree of confidence. As a regression model, knowing that our it can predict the fiber content of a recipe within an average range of about 1.5-1.7g is extremely useful. For reference, these (rounded) RMSE values can be reviewed in the R output below:

```
## [1] "RMSE against training data: 1.54"
```

```
## [1] "RMSE against testing data: 1.72"
```

Conclusion

In all, we were able to create a model that not only fit our given scenario, but produced decent results. While it does not predict fiber content perfectly, we do get within an average of about 1.6g of fiber when averaging out what we saw when working with the training and testing data. Utility for a model like this is two-fold.

For people navigating food recipes, there is the simplistic utility that one could predict fiber content and essentially isolate whatever recipes support one's own personal dietary needs. A more nuanced used-case for this model, however, would be as a generative guide. As in, savvy recipe creators could utilize a predictive model like this one to help guide the formulation of new recipes - aiming towards higher fiber content within their own recipes. While our model focused on predicting fiber content, the framework could easily be adapted to predict any of the other nutritional variables. In all cases, this has great utility as a means of promoting healthier eating habits for recipe consumers and producers alike.

Links and References

Alvin. (2020, December]). Food.com - Recipes and Reviews, Version 2. Retrieved November 12, 2021] from [https://www.kaggle.com/irkaal/foodcom-recipes-and-reviews/].

U.S. Department of Agriculture and U.S. Department of Health and Human Services. *Dietary Guidelines for Americans, 2020-2025*. 9th Edition. December 2020. Available at DietaryGuidelines.gov.

Code Appendix

Code from Introduction and EDA Sections

```
# Setup for overall RMarkdown environment.

knitr::opts_chunk$set(
  cache.extra = knitr::rand_seed,
  echo = FALSE,
  warning = FALSE,
  message = FALSE
  )

library(tidyverse)
library(e1071)
library(Metrics)

# The initial data was stored/read locally due to its original size.
# for the purpose of this report, the relevant objects were stored to/read from
# the github repo using saveRDS and readRDS.

recipes <- read_csv("./recipes.csv", show_col_types = FALSE)
reviews <- read_csv("./reviews.csv", show_col_types = FALSE)
```

```
# Function used to determine recipe rating rankings and prepare the data
# for downstream visualization.

get_ranks <- function(df, variable, order_levels, group_name = "rank") {
  df <- df %>%
    mutate(across(
      all_of(variable), factor, levels = order_levels, ordered = TRUE
      )) %>%
    group_by(across(all_of(variable))) %>%
    tally()

  if (variable != group_name) {
    colnames(df)[[1]] <- group_name
  }

  df$variable <- variable
  df
}

ratings <- seq(0, 5, 0.5)
review_ranks <- get_ranks(reviews, "Rating", ratings)
recipe_ranks <- get_ranks(recipes, "AggregatedRating", ratings)
```

```
# Generates recipe ranking visualization from EDA section.

recipe_col_charts <- bind_rows(review_ranks, recipe_ranks) %>%
  mutate(across(
    "variable",
```

```
    ~ if_else(
      . == "Rating", "Global Ratings (Actual)", "Available Subset (Aggregated)"
      )
    )) %>%
  ggplot(aes(rank, n)) +
  geom_col() +
  scale_y_continuous(labels = scales::comma_format()) +
  facet_wrap("variable", nrow = 2, scales = "free") +
  labs(title = "Food.com Recipe Ratings", x = "Rating", y = "Number of Recipes")


# Displays and saves the visualization to the local repository.
recipe_col_charts
ggsave("recipe_rankings.png")
```

```
# Function used to assess outlier boundaries for each numeric variable.

get_boundaries <- function(x) {median(x) + mad(x) * c(-3, 3)}

# Sets up a new subset dataframe including non-NA numeric variables.

recipe_dbl <- recipes %>%
  select(
    where(is_double),
    -AuthorId, -AggregatedRating, -DatePublished, -ReviewCount, -RecipeServings
    ) %>%
  drop_na()

# Applies previously obtained boundaries and filters the data set to a final
# overall subset used for sampling against.

boundary <- recipe_dbl %>%
  select(-RecipeId) %>%
  as.list() %>%
  map(get_boundaries)

recipe_dbl <- recipe_dbl %>%
  filter(
      between(
        Calories,
        boundary$Calories[[1]],
        boundary$Calories[[2]]
        ) &
      between(
        FatContent,
        boundary$FatContent[[1]],
        boundary$FatContent[[2]]
        ) &
      between(
        SaturatedFatContent,
        boundary$SaturatedFatContent[[1]],
        boundary$SaturatedFatContent[[2]]
        ) &
```

```
      between(
        CholesterolContent,
        boundary$CholesterolContent[[1]],
        boundary$CholesterolContent[[2]]
        ) &
      between(
        SodiumContent,
        boundary$SodiumContent[[1]],
        boundary$SodiumContent[[2]]
        ) &
      between(
        CarbohydrateContent,
        boundary$CarbohydrateContent[[1]],
        boundary$CarbohydrateContent[[2]]
        ) &
      between(
        FiberContent,
        boundary$FiberContent[[1]],
        boundary$FiberContent[[2]]
        ) &
      between(
        SugarContent,
        boundary$SugarContent[[1]],
        boundary$SugarContent[[2]]
        ) &
      between(
        ProteinContent,
        boundary$ProteinContent[[1]],
        boundary$ProteinContent[[2]]
        )
      )
```

```
# Generates the Fiber-variable visualization and
# saves it to the local repository.

fiber_plot <- recipe_dbl %>%
  ggplot(aes(FiberContent)) +
  geom_histogram(bins = 10) +
  scale_x_continuous(labels = scales::comma_format(1)) +
  scale_y_continuous(labels = scales::comma_format()) +
  labs(
    title = "Food.com Recipes by Fiber Content",
    x = "Fiber (g)",
    y = "Number of Recipes"
    )

fiber_plot
ggsave("fiber_plot.png")
```

```
# Generates correlation table for data set.

(recipe_cor <- cor(select(recipe_dbl, -RecipeId)))
```

Code from Data Preparation Section

```
# Conducts PCA and outputs summary of the prcomp object.

recipe_pca <- recipe_dbl %>%
  select(-c("RecipeId", "FiberContent")) %>%
  prcomp(scale = TRUE, center = TRUE)

summary(recipe_pca)
```

Code from Modeling Section

```
# The following function was used to run the optimization process
# with 5-fold cross validation and inform the selection of hyperparameter
# values for the end-model.

try_svm <- function(
  data, type, kernel, k = 5
  ) {
  tune.svm(
    FiberContent ~ PC1 + PC2 + PC3 + PC4 + PC5,
    data = data,
    type = type,
    cost = seq(0.01, 1, 0.01),
    tolerance = seq(0.001, 0.01, 0.001),
    gamma = seq(0.1, 0.2, 0.01),
    epsilon = seq(0.1, 1.0, 0.1),
    kernel = kernel,
    cross = k
    )
  }


# The following function was a simple wrapper used to evaluate the model
# against the typical performance metrics for regression models.

eval_regr <- function(data, model, new = FALSE) {
  if (new) {
    prediction <- predict(model, newdata = data)
  } else {
    prediction <- predict(model)
  }

  list(
    MSE = mse(data$FiberContent, prediction),
    RMSE = rmse(data$FiberContent, prediction),
    MAE = mae(data$FiberContent, prediction)
  )
}
```

```
# Given the outcome of our above optimization, the code in this chunk represents
# the end modeling procedure.

# A random seed is set.

set.seed(3)

# The PCA results are re-bound to the original recipe identifiers.

full_pca_data <- bind_cols(
  select(recipe_dbl, RecipeId, FiberContent),
  as_tibble(recipe_pca$x)
  )

# Distinct random samples are chosen from the overall data set, where the
# number of observations used were 1,000 with 80% (800) reserved for training
# and 20% (20) reserved for testing.

train <- slice_sample(full_pca_data, n = 800)
test <- slice_sample(anti_join(full_pca_data, train, by = "RecipeId"), n = 200)

# The following code is included, representing the usage of the optimization
# functions from the previous chunk. Kernel was selected manually via
# experimentation.

# svm_regr <- try_svm(train, type = "eps-regression", kernel = "radial")
# best_regr <- svm_regr$best.model

# The final model is fit below.

svm_regr <- svm(
  FiberContent ~ PC1 + PC2 + PC3 + PC4 + PC5,
  type = "eps-regression",
  kernel = "radial",
  data = train,
  cost = 1,
  gamma = 0.12,
  epsilon = 0.2,
)
```

```
# The following code outputs performance metrics against the training data.

regr_eval <- eval_regr(train, svm_regr)
```

```
# The following code outputs performance metrics against the test data.

regr_eval2 <- eval_regr(test, svm_regr, new = TRUE)
```