Capstone 1 Project Report

by Anna Kantur

date: 03/21/2020

Contents

i. Problem Statement

This project was inspired by a personal interest in food and cooking and trying to answer the question of what successful recipes have in common. Is it nutrition value, number of steps required, time required, a specific ingredient, or something else?

Answers to these questions will interest various groups from grocery store owners who decide what products to put on display for advertisement to professional and amateur cooks who are trying to create the best recipes.

ii. Data Collection and Wrangling Summary

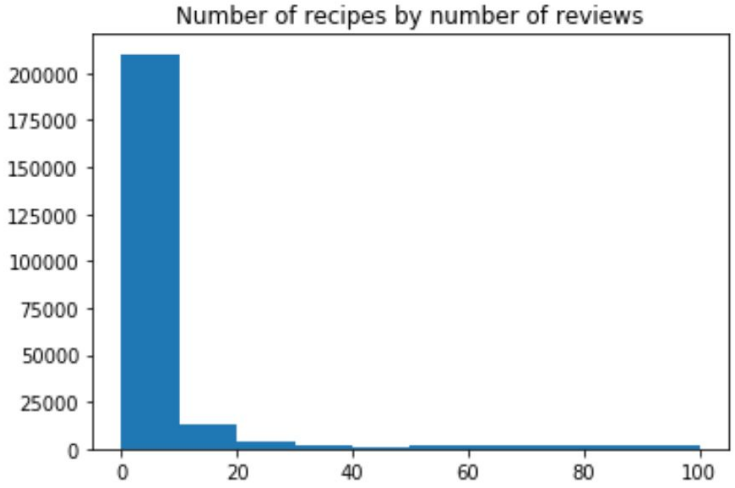
Data Collection

We collected the data from Kaggle:
https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions

This dataset consists of 180K+ recipes and 700K+ recipe reviews covering 18 years of user interactions and uploads on Food.com (formerly GeniusKitchen). The dataset was previously used in the research paper by Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, Julian McAuley called “Generating Personalized Recipes from Historical User Preferences”. We use the 'RAW_recipes.csv' and 'RAW_interactions.csv' files and upload them into Pandas data frames.

Irrelevant Data and Missing Values

For the purposes of this Capstone 1 project we consider recipes with less than 10 reviews outliers. We explore the presence of recipes with less than 10 reviews by using the histogram plot from matplotlib and by filtering the pandas data frame by the number of recipes that had reviews.



The original interactions data frame has 210,244 recipes, but out of them there are only 18,762 recipes with more than 10 reviews. We update the recipes and reviews data frames to only include recipes with over 10 reviews.

Upon investigation of the recipes data frame we also discover that the description column has 406 empty values. We will keep the recipes with missing descriptions in the data for now.

Average Rating per Recipe

We analyze the average review rating and the number of users who reviewed by recipe in the resulting data frame. Most of the recipes have a review rating of 3.5 or higher

0 Star Ratings

0 star ratings are present in 58% of unique recipes and represent 6% of all the reviews in the reviews dataframe. By reading the reviews for 0 star ratings it is evident that some of 0 star reviews were meant to have a higher rating and were probably coded as 0 star by error. Nevertheless, we check that food.com, where the reviews were collected, allows 0 star ratings. So we will keep 0 star ratings in the reviews dataframe.

Recipe Tags

First, we investigate the tags column of the recipes data frame. Tags are single words or simple expressions that recipe posters add to a recipe on the website for ease of discovery. The tags column data in the recipes data frame is presented so that the individual values look like a list, but are actually strings. Example of tags for one recipe:

["'weeknight'", " 'time-to-make'", " 'course'", " 'main-ingredient'", " 'preparation'", " 'main-dish'", " 'pork'", " 'crock-pot-slow-cooker'", " 'dietary'", " 'meat'"]

There are 495 unique tags from all the recipes.

We create a tags_matrix data frame with recipe ids in rows, all individual tags as columns, and values being 1 if the ingredient is present in the recipe and 0 if not. To do this, we perform the following steps:

1. Add a column tags2 to the recipes data frame where the values from the tags column are converted to a list
2. Add a column tags_check to the recipes data frame with a list of unique tags from all the recipes

3. Create a function that runs through all the values in the tags_check column and replaces the tag value with 1 if it is present in the recipe and with 0 if it is not present. We test that the function is working on one recipe id.

Recipe Ingredients

We investigate the ingredients column of the recipes dataframe. The data looks like a list, but the type of data is actually a string. We create a list with 8,091 unique ingredients from all the recipes.

We discover that this list has similar ingredients with different names. We choose to not modify the ingredients column itself and address the issue further by data wrangling the ingredients matrix.

We narrow down the list of the ingredients to only the ones that are present in more than a 100 recipes. There are 245 of such ingredients.

The ingredients matrix is created similar to the tags_matrix. We also clean up the ingredients_matrix to get rid of:

- ingredients with names in multiple (e.g. ‘eggs’ vs ‘egg’)
- ingredients with the same first word in the name (e.g. ‘garlic powder’ vs ‘garlic’)
- ingredients with the same second word in the name (e.g. ‘fresh cilantro’ vs ‘cilantro’)
- ingredients with the same third word in the name (e.g. ‘freshly ground pepper’ vs ‘pepper’)
- We make sure that the values from the removed columns are correctly moved to the relevant ingredient column in the ingredients_matrix. E.g. if ‘garlic powder’ column has 1 and ‘garlic’ column has 0 for a particular recipe, then the unified ‘garlic’ column has a 1.

The resulting ingredients_matrix had 149 columns.

Files for further use in Capstone 1

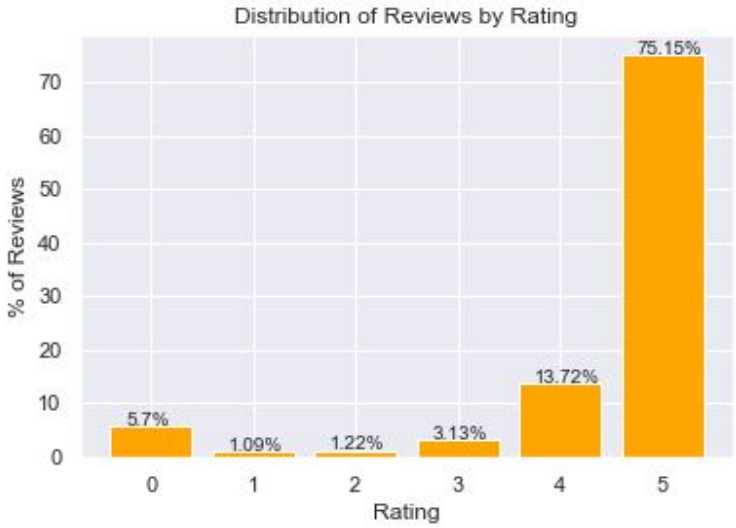
As the result of data wrangling we have several files for further analysis: 1) recipes_df, 2) reviews_df, 3) tags_matrix, 4) ingredients_matrix.

We save these files in csv format to use in another Jupyter notebook for data story.

iii. Exploratory Data Analysis

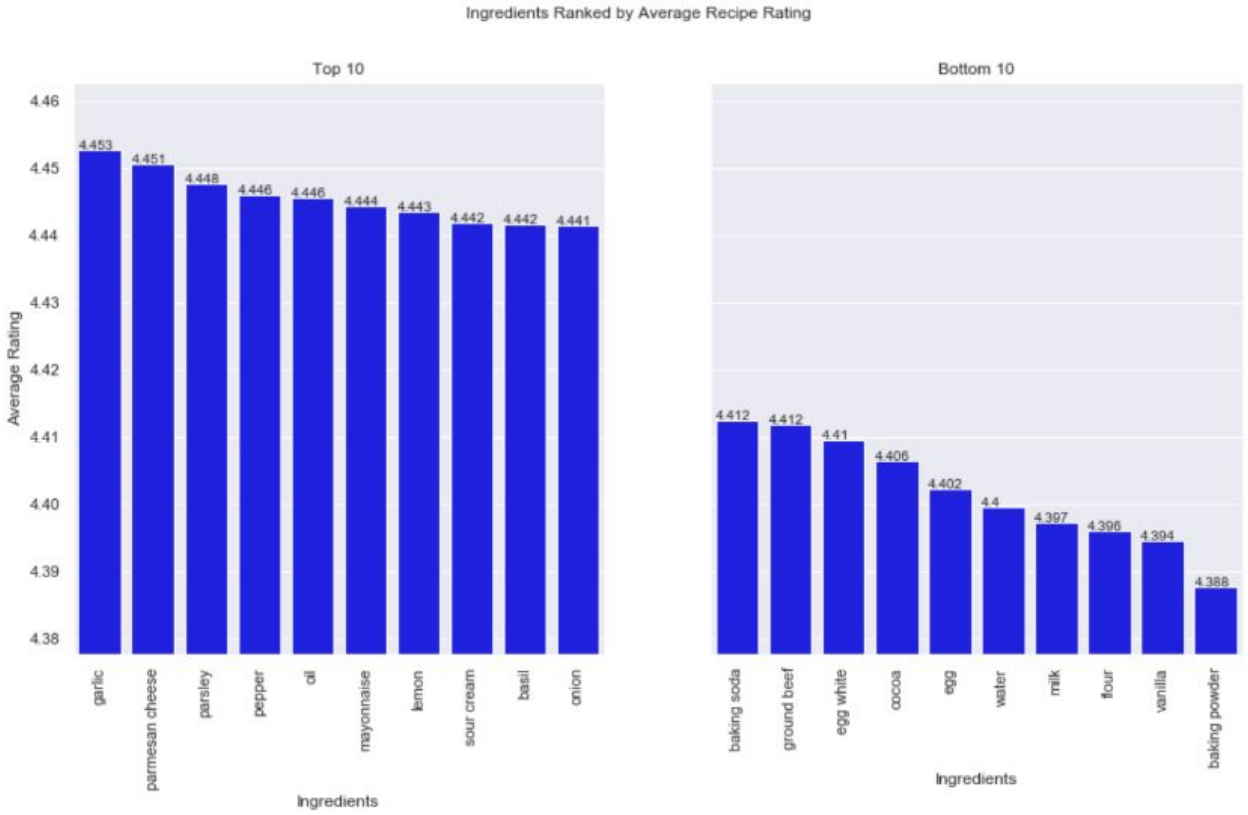
Average recipe rating across all reviews

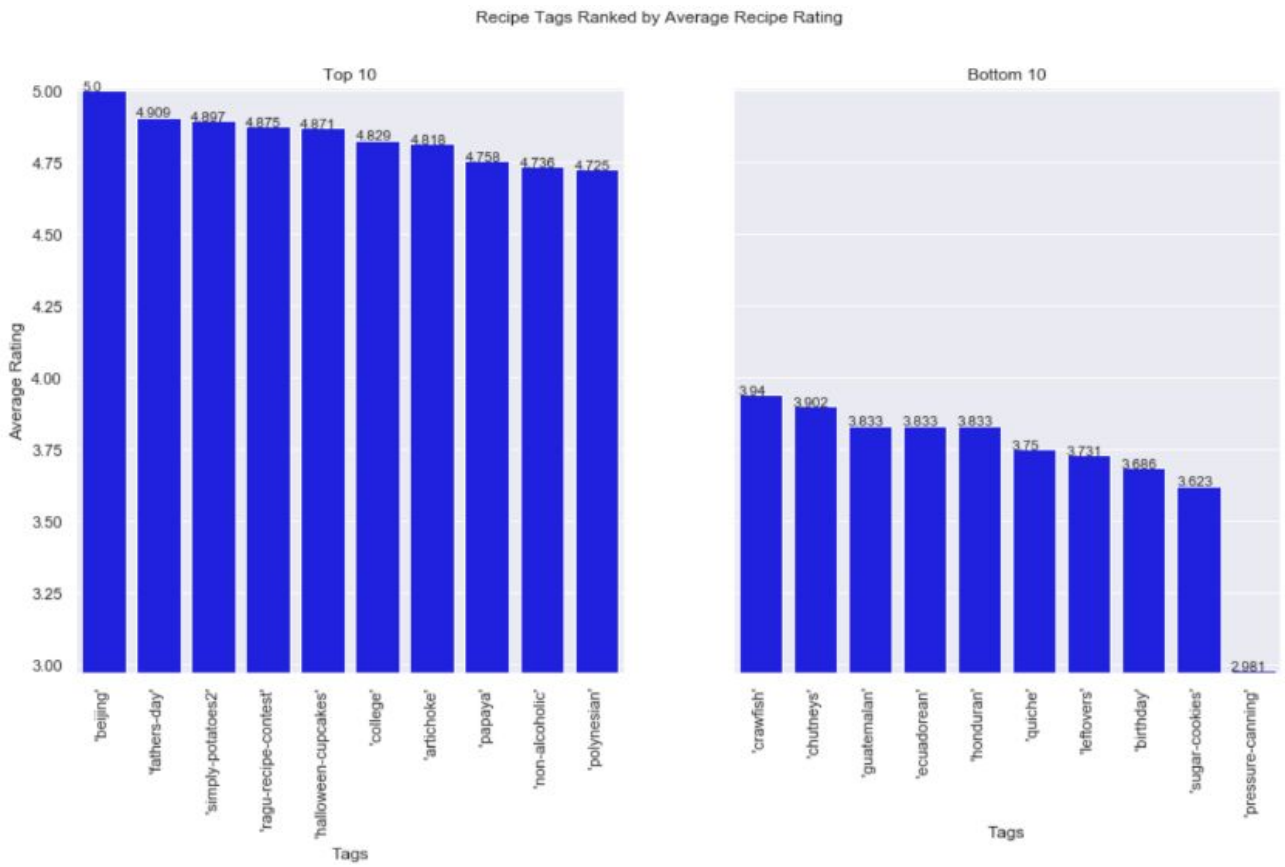
Average recipe rating across all reviews is 4.435.



Top 10 Ingredients and Tags by Average Recipe Rating

We identify the best and the worst ingredients and tags by recipe rating.



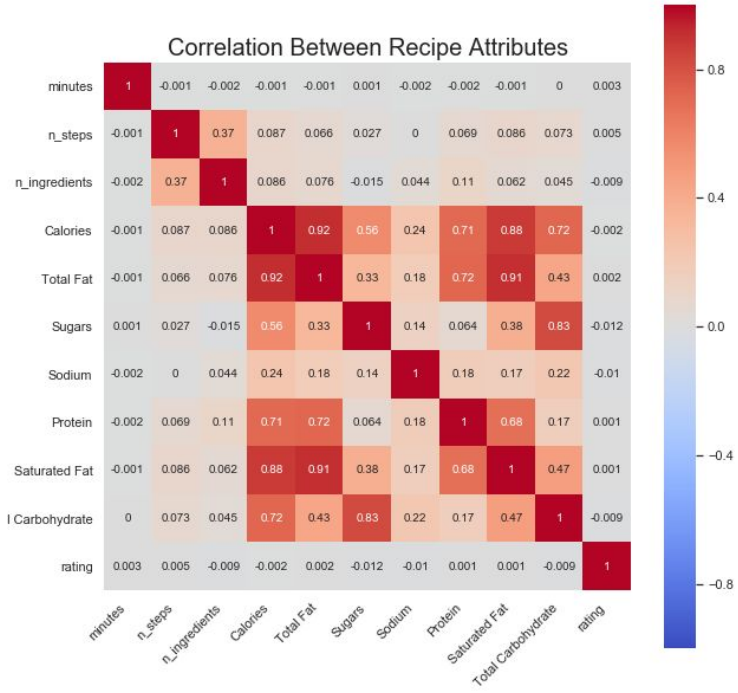


The best ingredient is garlic. It has the average recipe ranking of 4.453. The worst ingredient is baking powder. It has the average recipe ranking of 4.388. The best tag is 'beijing'. It has the average recipe ranking of 5.0. The worst tag is 'pressure-canning'. It has the average recipe ranking of 2.981. (As a reminder,t we are looking at recipes with over 10 reviews, and we know from the previous analysis that the majority of such recipes have an average rating of 3.5 or higher. Also, we are only analyzing the ingredients present in 100 or more recipes.)

Correlation Matrix

We also show the nutritional values correlation matrix.

The correlation matrix shows that there is a strong correlation between calories content and total fats (saturated fats specifically), protein and carbohydrates. However, there is no strong correlation between the recipe rating and any of the mentioned recipe attributes.



The Best and Worst Ingredients and Tags

We further investigate the data in EDA and discover the following findings:

1. The best ingredient is garlic. It has the average recipe ranking of 4.453
2. The worst ingredient is baking powder. It has the average recipe ranking of 4.388
3. The best tag is 'beijing'. It has the average recipe ranking of 5.000
4. The worst tag is 'pressure-canning'. It has the average recipe ranking of 2.981
5. Successful recipes on average take longer to make and include 10 or more steps (not statistically significant as shown below)
6. In terms of nutritional value, successful recipes have more total fats, sugars and carbohydrates, but less sodium and saturated fats, and slightly less protein.

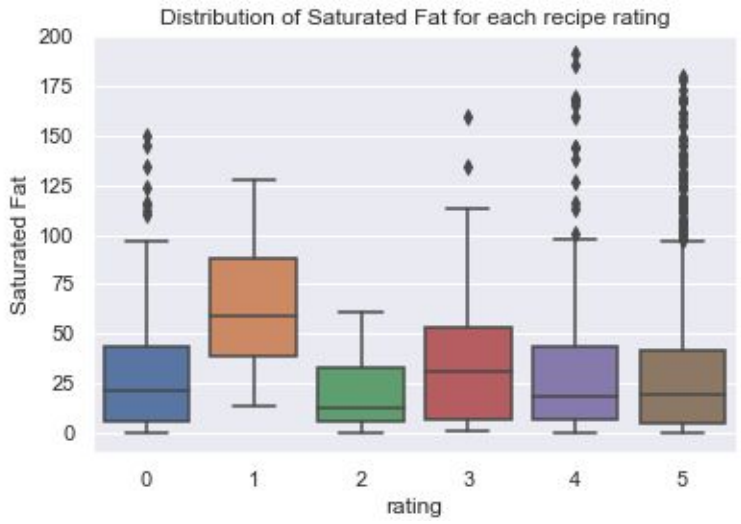
T-Tests

We perform a two sample t-tests to confirm our EDA findings. The two independent samples are the recipes with a specific attribute and the recipes without it. The null hypothesis $H0$ is that the mean recipe rating for the two samples is identical. The alternative hypothesis Ha was that the rating means are different (e.g. the mean recipe

rating is affected by a specific recipe attribute). If the t-test results were statistically significant (e.g. p-value > a, a = 0.05), then we rejected the $H0$ and accepted the Ha.

t-test results:

1. **Garlic** The test had p-value of 9.475783077096418e-18 which is very small, so we can reject the $H0$ and confirm that garlic in recipes contribute to a higher recipe rating
2. **Baking Powder** The test had p-value of 9.475783077096418e-18 which is very small, so we can reject the $H0$ and confirm that baking powder in recipes contribute to a lower recipe rating
3. **'Beijing' recipe tag** The test had p-value of 0.13 > a of 0.05, so we can reject the Ha. This means that 'beijing' tag positive effect on the recipe rating is statistically not significant.
4. **'pressure-canning' recipe tag** The test had p-value of 2.792736995754129e-10 which is very small, so we can reject the Ho. This means that 'pressure-canning' tag negative effect on the recipe rating is statistically significant
5. **Saturated Fats Content** was the only other attribute that affects the recipe rating in statistically significant way (the test p-value was 0.03). All the other recipe attributes did not significantly affect the recipe rating.

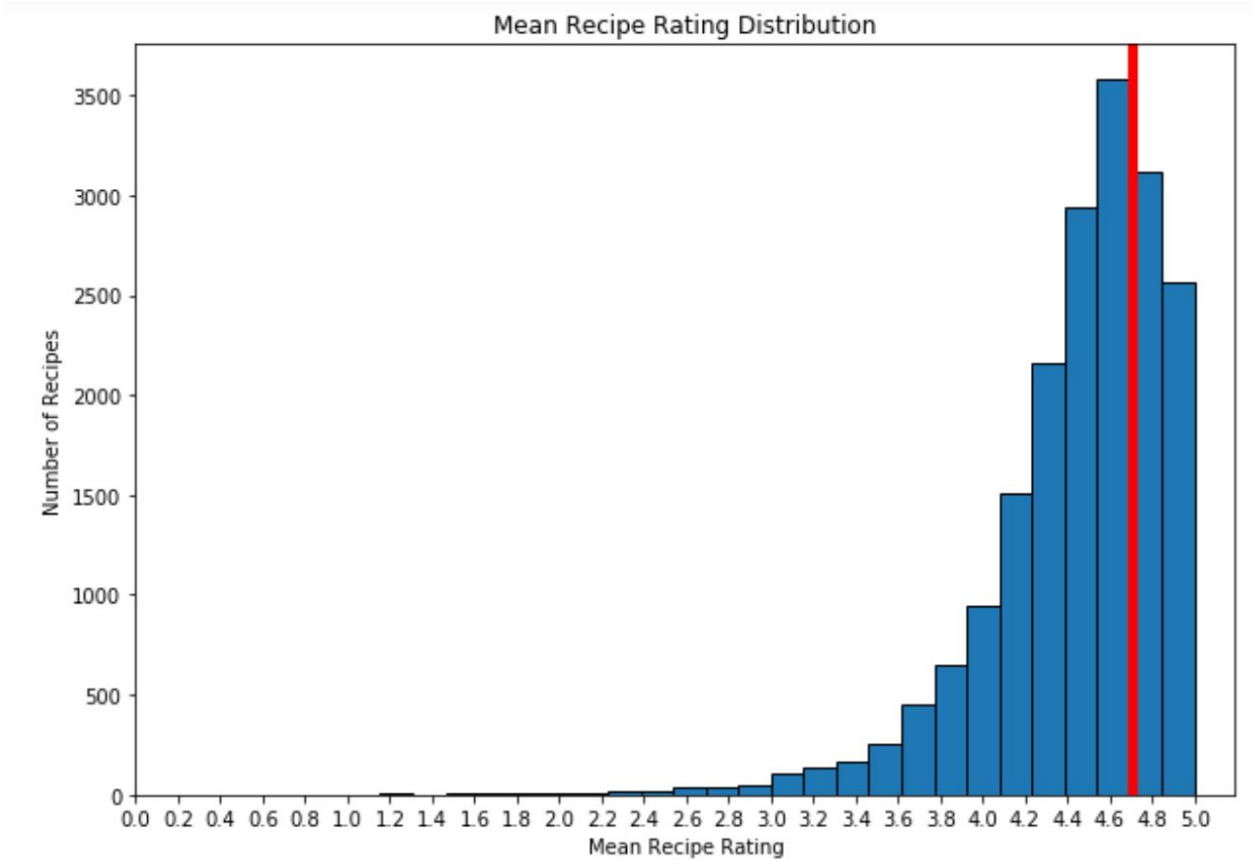


iv. In-depth Analysis Using Machine Learning

Predictive Modelling

We will now dive deeper and build a model to classify recipes as “good” or “bad” by mean recipe rating.

We choose 4.7 average recipe rating as a divide for “good” vs “bad” recipes:



There are 5,519 “good” recipes and 13,302 “bad” recipes in our dataset.

Choosing the Best Model

We will try Random Forest, Logistic Regression and KNN classifier models. We use the same approach for each of the three models:

1. Build the model using sklearn package
2. Tune in hyperparameters
3. Estimate the ROC_AUC score

In addition, for Random Forest we find top 20 best features by using feature_importances method on the classifier. Here is the list of top 20 most important features by the importance score:

0 ('Calories', 0.03599724473589182)

1 ('Sugars', 0.034046714261403835)

2 ('Protein', 0.032015894747727876)

3 ('minutes', 0.031239016224030726)

4 ('Saturated Fat', 0.03111097618770412)

5 ('Sodium', 0.03047736900924014)

6 ('Total Fat', 0.030016450811690142)

7 ('Total Carbohydrate', 0.028400197936175275)

8 ('n_steps', 0.02699280475047596)

9 (" 'easy'", 0.007238298792856107)

10 (" 'equipment'", 0.006913044424659004)

11 (" 'occasion'", 0.006421651924366628)

12 (" 'number-of-servings'", 0.0061877163666479535)

13 (" 'cuisine'", 0.006035790961877401)

14 (" 'oven'", 0.005836153491809918)

15 ("'60-minutes-or-less'", 0.005774535718058847)

16 (" 'low-in-something'", 0.0056769836799905275)

17 ('salt', 0.005489661715824159)

18 (" 'taste-mood'", 0.005294530153120957)

19 (" 'inexpensive'", 0.005267690999395954)

Turns out that all the nutritional values and time to cook the recipe ("minutes", "steps", "easy", "60-minutes-or-less") are in the top 20 most important features to predict the goodness of the recipe average rating. We rerun the Random Forest model with the best features to see if we can get a score improvement.

The summary of all the models that we try:

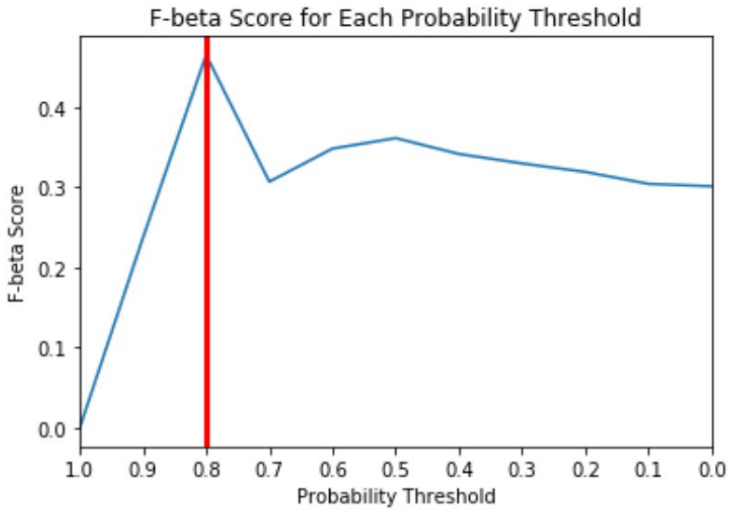
Model	**Best Score**	**Best Parameters**
Random Forest	62%	{'max_depth': 11, 'max_features': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0, 'n_estimators': 100}
Random Forest with 20 Best Features	57%	{'max_depth': 7, 'max_features': 5, 'min_samples_leaf': 3, 'min_samples_split': 5, 'min_weight_fraction_leaf': 0, 'n_estimators': 100}
Logistic Regression	54%	{'C': 8.483428982440725e-05, 'class_weight': None}
KNN	54%	{'n_neighbors': 20, 'weights': 'distance'}

The best ROC_AUC score is for the Random Forest model (61%), so this is the best model to predict the average recipe rating being good or bad.

Thresholding Probabilities by the Best F-Beta Score

The default probability threshold is 0.5, but we decide to challenge that.

1. We loop over the probability thresholds from 0 to 1
2. We estimate the probabilities of predicting class 1 (the “good” recipes)
3. We transform the probabilities from the previous step into 1s and 0s, based on whether the estimated probability is higher or lower than the threshold probability
4. We use our ultimate metric: F-beta score with beta < 1 to be weighted towards precision, - to pick the best probability threshold



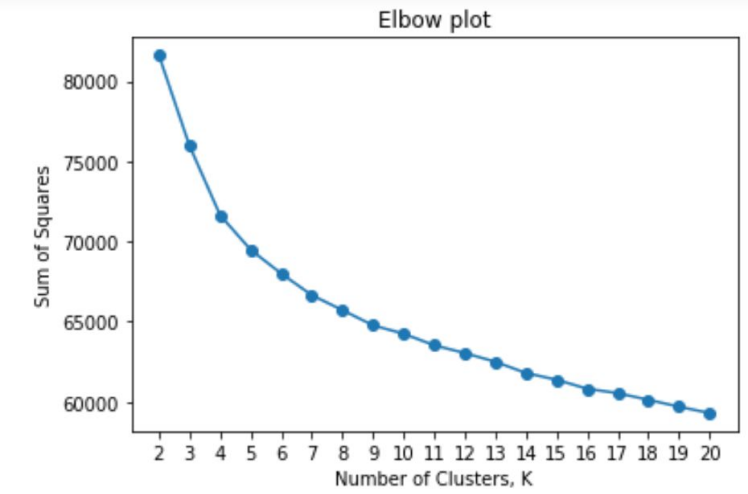
A chart above shows probability thresholds on X axis and F-beta score on Y axis. The vertical red line represents the optimal threshold of 0.8, at which the F-beta score is 46%.

Clustering

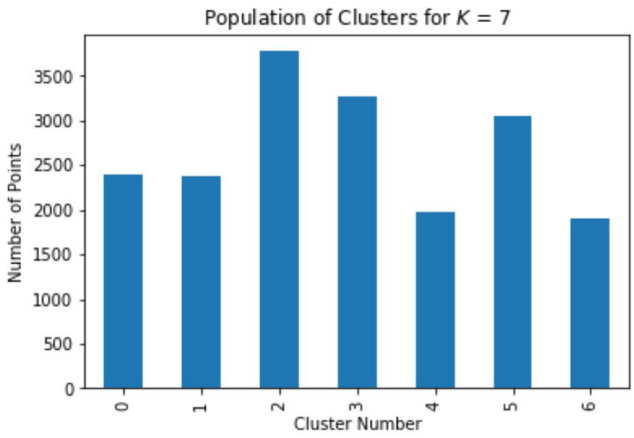
We use ingredients data to cluster recipes into groups and see if there are any patterns. We build a K-Means Clustering model and choose the number of clusters K by using the Elbow Method and Silhouette method.

The Elbow Method

We construct an Elbow plot showing the sum-of-squares for each K between 2 and 20:



From the chart above we infer that the best K is somewhere between 3 and 7. We try K=7 and make a bar chart showing the number of points in each cluster for K-means under the best K:



The Silhouette Method

We decide to use a second method for selecting K-value: the Silhouette method. This method measures how well each datapoint "fits" its assigned cluster and also how poorly it fits into other clusters. This is a different way of looking at the same objective. We need to choose K with the silhouette score closest to 1.

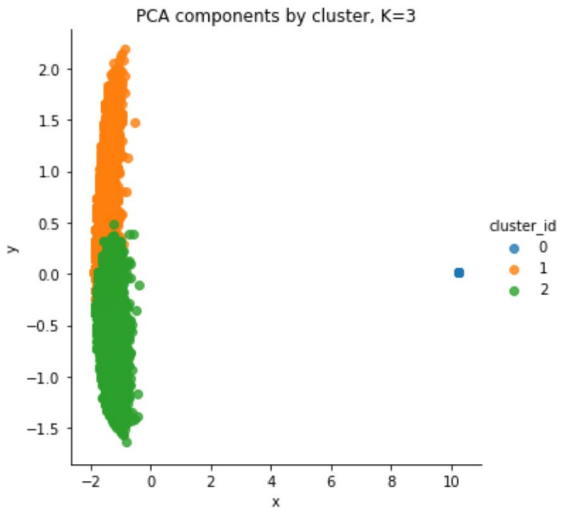
For K between 4 and 7 we get the following results:

- For n_clusters = 3 The average silhouette_score = 0.19172585856463387
- For n_clusters = 4 The average silhouette_score = 0.18957449729577003
- For n_clusters = 5 The average silhouette_score = 0.18111405590654364
- For n_clusters = 6 The average silhouette_score = 0.16267324366196434
- For n_clusters = 7 The average silhouette_score = 0.1632525460602517

The silhouette score is best for K=3. And even then the score is below 0.25, which indicates that no substantial structure has been found. We decide to use K=3 as the best K for K-means clustering.

PCA Dimension Reduction

To visualize clusters, we use PCA to reduce the dimensionality of our data from 149 ingredients (dimensions) to 2 dimensions:



To summarize what we did so far, we took the columns of 0/1 for ingredients and transformed them into a 2-D dataset. We took one column and arbitrarily called it x and then called the other y. We showed the x and y points on a scatterplot. We color coded each point based on it's cluster so it's easier to see them. As seen from the scatterplot, there was a clean break between the clusters, and the clusters did not overlap a lot.

Resulting Clusters

We observe the data points in 3 clusters:

	recipe_name	top_10_ingredients	Calories	Sugars	Sodium	Protein	Saturated Fat	Total Carbohydrate
0	[da best chicago style italian beef, i hate m...	[sesame seeds, thyme, ginger, tabasco sauce, f...	1285145.0	297944.0	74806.0	77959.0	119762.0	48137.0
1	[chile rellenos, chinese candy, healthy for t...	[sugar, salt, butter, egg, flour, milk, water,...	3850483.8	911326.0	226026.0	224000.0	358662.0	148227.0
2	[chicken lickin good pork chops, grilled ve...	[salt, garlic, onion, pepper, oil, butter, wat...	3384196.1	224985.0	322173.0	365498.0	329119.0	79829.0

Cluster 1 is the most obvious. The recipes have the highest sugar and carbohydrates content. By observing the recipe names it seems like there are a lot of desserts and sweet things in this cluster.

Cluster 0 seems to have a combination of desserts, salads and meat dishes. It is the lowest cluster by all nutritional values, so we conclude that this is a "mid-day meal" cluster.

Cluster 2 has a lot of recipes with meat and savory recipes, and has the highest protein and sodium content. We conclude that this is the "Dinner" cluster.

To summarize, while no substantial cluster structure was found, we can classify the 3 clusters as:

1. Desserts/Baking Cluster
2. Mid Day Meal Cluster
3. Dinner Cluster

Recommendations

There are several recommendations that result from our analysis:

1. Use garlic in recipes, people love it
2. Avoid recipes requiring pressure-canning or high amounts of saturated fats
3. For grocery store owners it might be wise to arrange the products around the three recipe clusters that we found: Desserts/Baking Cluster, Mid Day Meal Cluster, and Dinner Cluster
4. To get an idea of how much people would like your recipe, think about nutritional values and how long it takes to make the recipe

Ideas for Further Research

It is worth mentioning that no significant recipes cluster structure has been found, and even the best classification model has a very low ROC_AUC score of 62% (50% being the minimum). Also, Python gives a warning about F-beta score being ill-defined. So further exploration and adjusting the border for “good” and “bad” recipes by average score can uncover more meaningful insights into data.