Springboard Data Science Intensive Program

Capstone Project 2:

Building a Collaborative Filtering Recipe Recommendation System

By: Julien Laks

September 2020

Table of Contents

I. Introduction and Data Description

The goal of this project is to explore potential improvements that could be made to the well-known recipe-sharing website, all-recipes.com to increase both user-satisfaction and revenue. In particular, we will focus on the following business goals :

Goal 1 : Group Recipes into different categories using recipe ingredients, then segment website users into different groups according to their recipe preferences. The idea behind goal is that knowing more about the website's users may help to improve some strategic decisions. For example, if the website decided to partner with a brand selling ingredients or cooking utensils, it would be of great value what kind of recipes each customer likes the most so as to better target the marketing campaign.

Goal 2 : Build a personalized recommender system for frequent users of the website.The benefit of a good recommender system is quite obvious : if the website is able to recommend to each user the recipes that best suits their taste, users will be more likely to come back to the website.

The Data:

The data was downloaded from Kaggle : https://www.kaggle.com/kanaryayi/recipe-ingredients-and-reviews?select=clean_reviews.csv

It is made of two distinct datasets. The first dataset contains basic information about about 40000 recipes from allrecipes.com , in particular, each recipe is described by the following variables:

`Recipe_ID` : Unique recipe identifier
`Recipe_Name` : Name of recipe as given on the website
`Ingredients` : List of ingredients, string format
`Instructions` : Instructions, sting format

The second dataset provides information about user interactions, and contains the following variables :

`Reviewer_ID` : Unique user identifier
`Recipe_ID` : Unique Recipe identifier
`Rating` : The rating given by the user to the recipe

II. Data Preprocessing and feature extraction

The following example shows how the ingredient for each recipe were listed:

```
['1/2 cup unsalted butter, chilled and cubed', '1 cup chopped onion', '1
3/4 cups cornmeal', '1 1/4 cups all-purpose flour', '1/4 cup white sugar',
'1 tablespoon baking powder', '1 1/2 teaspoons salt', '1/2 teaspoon baking
soda', '1 1/2 cups buttermilk', '3 eggs', '1 1/2 cups shredded pepperjack
cheese', '1 1/3 cups frozen corn kernels, thawed and drained', '2 ounces
roasted marinated red bell peppers, drained and chopped', '1/2 cup chopped
fresh basil']
```

The list of ingredients not only contained ingredient names, but also quantities and comments. In order to be able to group recipes into different categories based on the list of ingredients contained in each recipe, we first need to extract ingredient names from the ingredient list, such as the above list is transformed into the following :

```
['buttermilk','corn','basil','baking soda','white sugar','flour','egg',
'cornmeal','baking powder','unsalted','marinated red bell','onion','salt',
'cheese']
```

This ingredient-extraction task was achieved by using an LSTM based Named Entity Recognition (NER) model. The model was trained on the New York Times Tagged Ingredient Dataset, which contains a list of 17000 recipe ingredient lists in which the words in each ingredient list have been tagged according to whether they represent an ingredient, a quantity, a unit, or a comment :

	index	input	name	qty	range_end	unit	comment
0	0	1 1/4 cups cooked and pureed fresh butternut s...	butternut squash	1.25	0.0	cup	cooked and pureed fresh, or 1 10-ounce package...
1	1	1 cup peeled and cooked fresh chestnuts (about...	chestnuts	1.00	0.0	cup	peeled and cooked fresh (about 20), or 1 cup c...
2	2	1 medium-size onion, peeled and chopped	onion	1.00	0.0	NaN	medium-size, peeled and chopped
3	3	2 stalks celery, chopped coarse	celery	2.00	0.0	stalk	chopped coarse
4	4	1 1/2 tablespoons vegetable oil	vegetable oil	1.50	0.0	tablespoon	NaN

The trained model NER showed an validation accuracy of about 99.3% on the New York Times Dataset.

I then applied this trained model on the recipes dataset, to create a new feature containing the list of unique ingredients contained in each recipe.

III . Exploratory Data Analysis

Our data contains information about 48640 recipes, with 3787088 ratings made by 1158858 users. The great majority of users have rated only very few recipes : 87% of the users rated 5

recipes or less, and 97% of the users rated 20 recipes or less. There are, however, a few "super-raters" among the users who have rated up to 500 recipes.

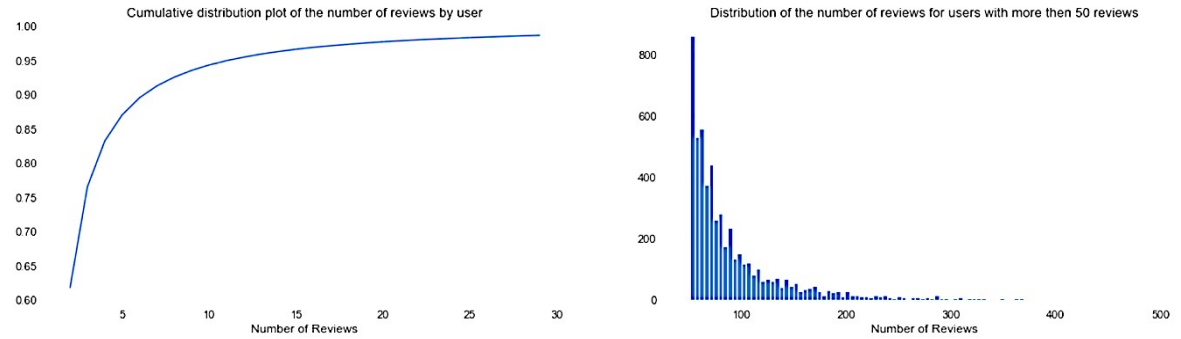


Figure 1. Distribution of Recipe Reviews

Most of the users tend to give very good ratings to the recipes they rate : out of a scaled of 1 to 5 stars, more than 50% are 5 stars, and only 10% of all ratings are 3 stars or less.

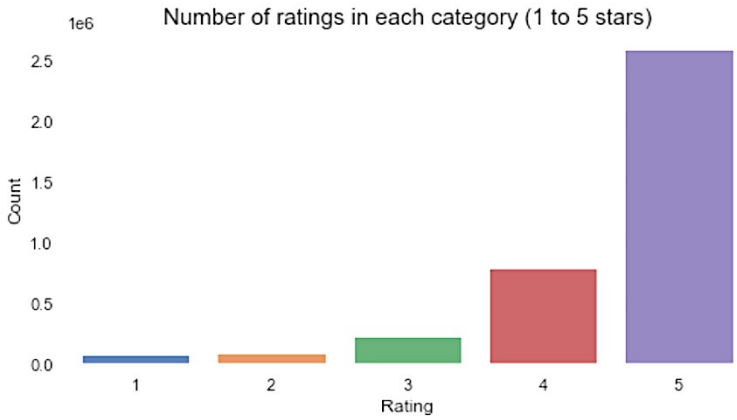


Figure 2. Distribution of Recipe Ratings

Does this mean that people are generous about the ratings they give to recipes ? This is certainly true to some degree, however, the following plot shows that recipes which are badly rated also receive very few ratings. From we can infer that users are not willing to try out recipes that other users rated badly. In other words, only "good" recipes seem to attract many users, and because these recipes are "good", users also give them good ratings. This also seems to point out the interesting fact that unlike for books or movies, people tend to have very similar tastes regarding food.

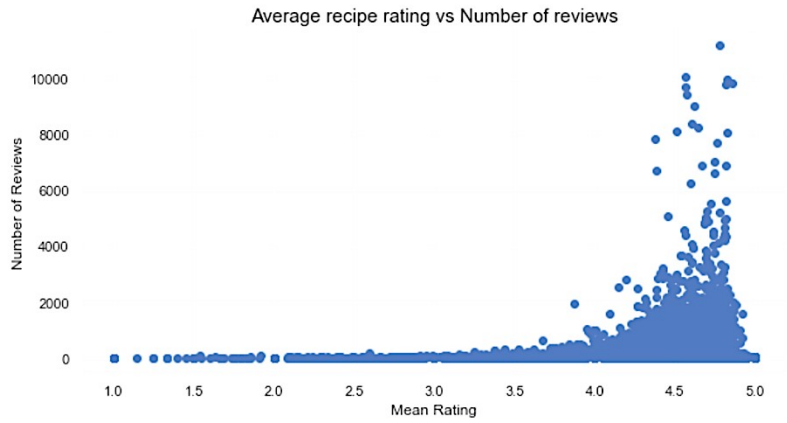


Figure 3. Number of Reviews vs. Mean Rating

That being said, it is also clear from the plot below that some users are more severe than others in their ratings, with some user giving as many 4 star as 5 star ratings, whereas a significant proportion of users give almost only 5 stars ratings

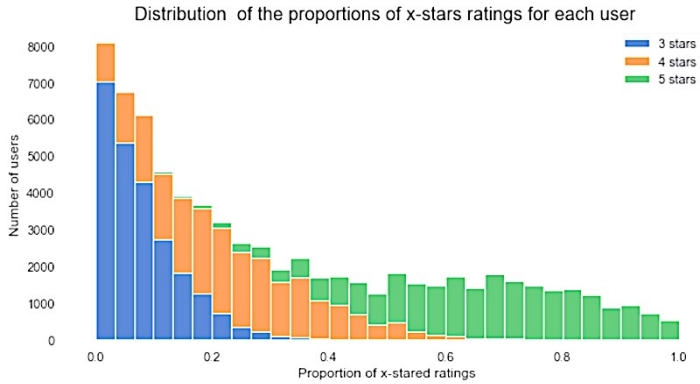


Figure 4. Rating Variability among users

Let us now have a look at the picture of some of the most popular (recipes with most ratings), and some of the least popular (recipes with least ratings and low ratings) recipes.
Interestingly, the recipes depicted in the the images of the most popular recipes in fact do look tastier than the less popular ones. It would therefore be interesting to build a computer vision model able to predict whether or not a recipe is likely to be popular or not. This is however not the object or our present project.

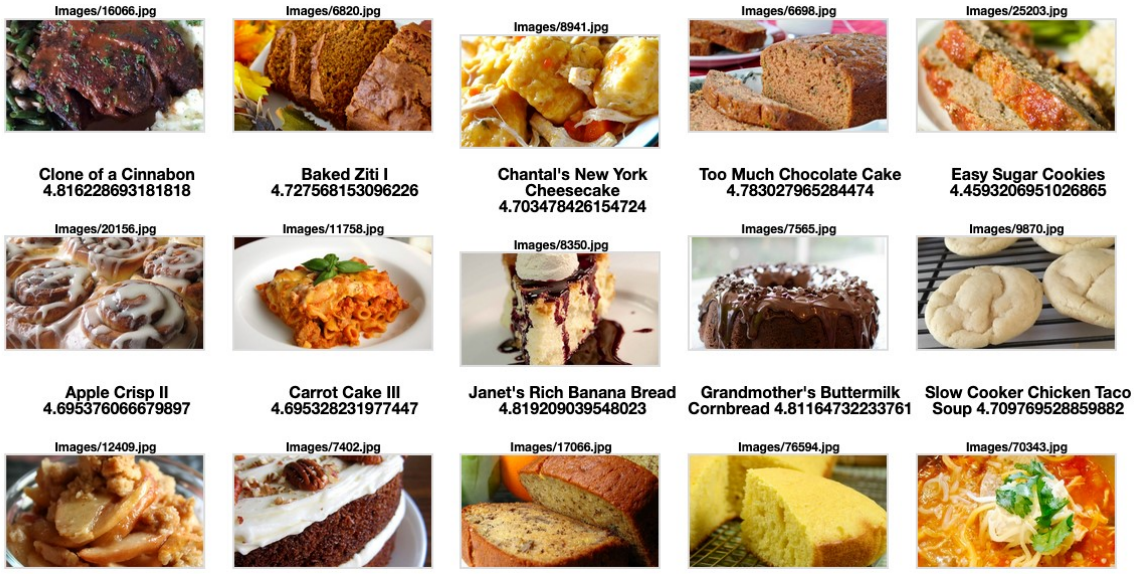


Figure 5. Most Popular Recipes

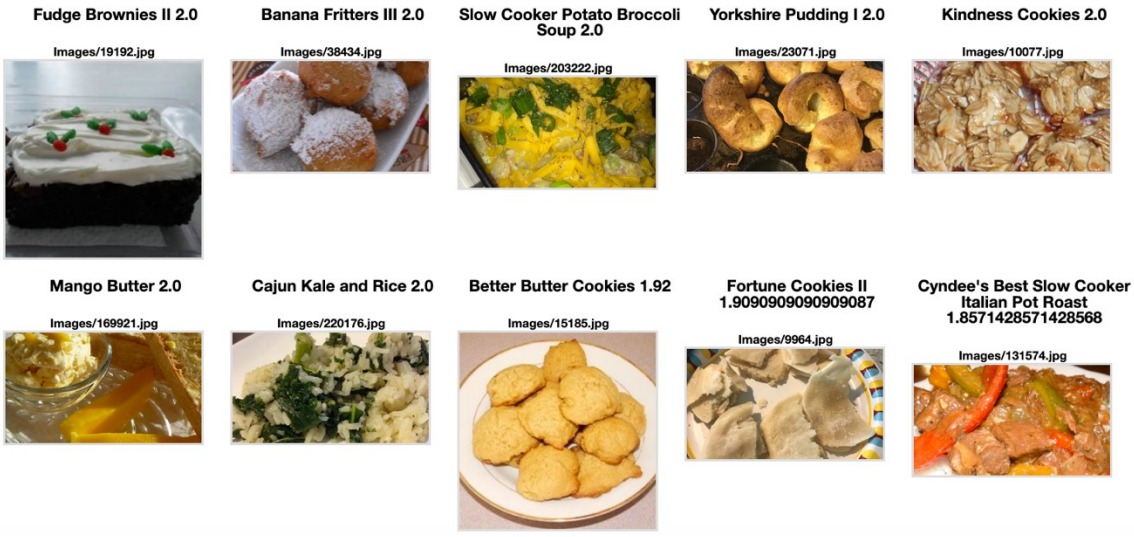


Figure 6. Least Popular Recipes

One might also wonder whether or not the list of ingredients contained in a recipe is correlated to the recipe's global rating. In order to answer this question, I proceeded in two steps. First, for each ingredient I computed the ingredient's "rating" by taking the average rating of all the recipes in which the ingredient appears. Then, for each recipe, I computed the average rating of all ingredients contained in the recipe. The figure below shows us that in fact, some ingredients are likely more likely than others to improve the recipe's quality. Among the best rated ingredients, we find :

```
orange bell pepper, potato starch,peach schnapps,pomegranate seed ,creme
fraiche, crawfish tail,bar chocolate,espresso, powder ,pickle ,collard
green,lime wedge ,mexican oregano
```

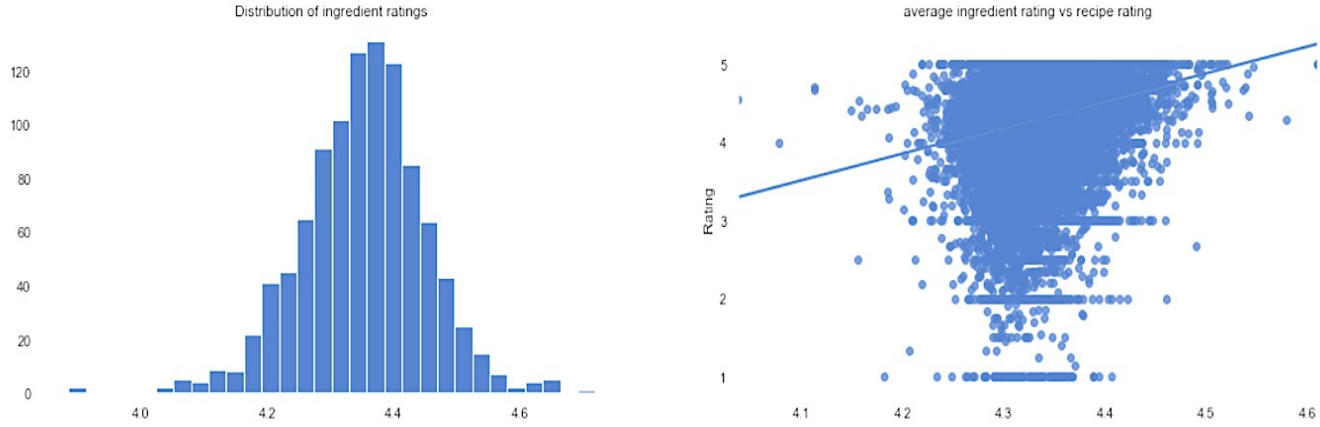


Figure 7. Ingredient Ratings

It is doubtful, however, that the sheer list of ingredients will be enough to predict any recipe's quality, as many other factors, such quantity, combinations, cooking methods, difficulty, are also crucial.

IV. User segmentation based their on recipe preferences

We now present our findings regarding the first goal of this project, namely clustering users into different categories, depending on what kind of recipes they like the most.

a/ Categorizing recipes

In order to be able to group users into different customer segments, we first need to somehow categorize recipes, so that each user's choices actually inform us about what kind of recipes they like.

In order to group recipes into different clusters, I proceeded in two steps. First, I divided the recipes into different groups according to the last word in the recipe's name. In fact, as the following sample list clearly shows, the last word of the title is the most informative single element of each recipe :

```
'cornbread', 'twist', 'roll', 'bread', 'bun', 'muffin', 'doughnut',
'biscuit', 'pretzel', 'loaf', 'dough', 'scone', 'iii', 'waffle', 'machine',
'tortilla', 'spread', 'pinwheel'
```

After filtering out the words that appear in less than 10 recipe names, I obtained a list of 1585 recipe title category Next, using a tf-idf combined with a cosine similarity metric on the list ingredients that appear most often in each of these categories, I was able to narrow down the recipe categories into 20 different recipe groups.

Group 17: `'coffee', 'drink', 'granola', 'latte', 'milkshake', 'parfait', 'shake','trifle'`

Group 10 : `'burrito', 'dip', 'enchilada', 'jalapeno', 'nacho', 'popper', 'quesadilla', 'wonton'`

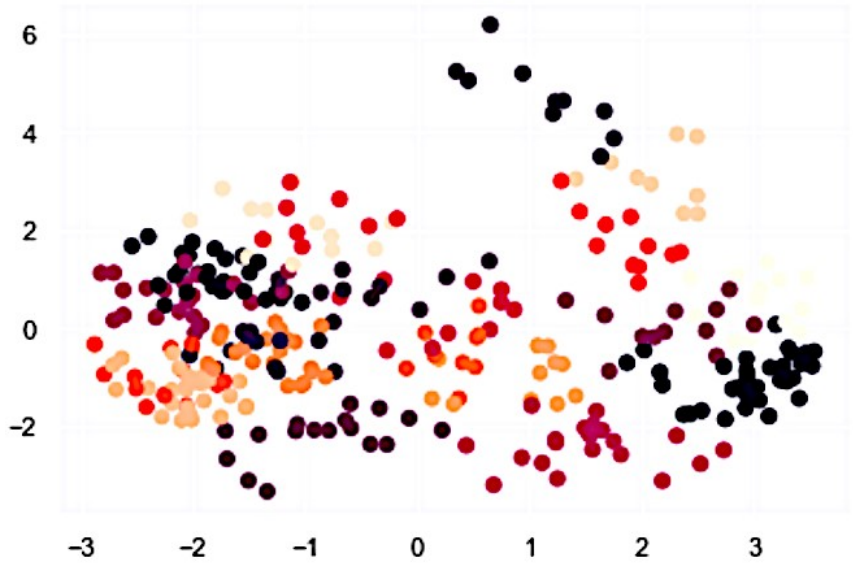
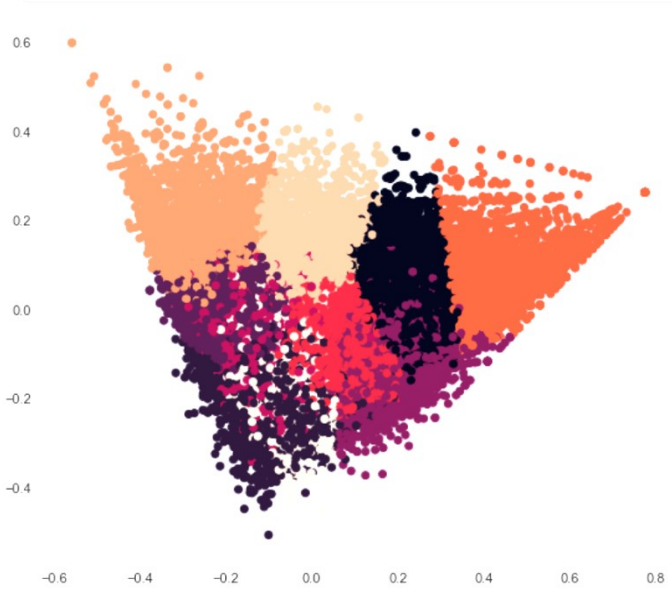


Figure 8. Recipe Groups (2D representation with PCA)

Then, based on how many recipes in each category every user liked, I computed the similarity between each pair of users, and finally I created 10 different user segments based on their taste similarity.

Figure 9. User Segments



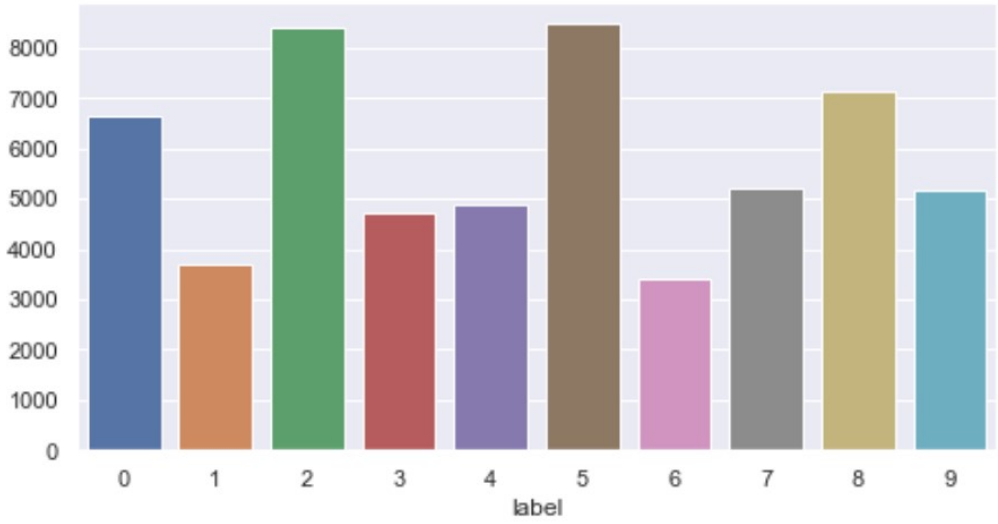


Figure 10. Distribution of Users in each Segments

We observe a clear difference in preferences between users in each category (see figures below). For example, Users in category 4 prefer recipes with a lot of meat, whereas category 3 users like creamy desserts, and category 8 love salads and soups : this tells us that our user categorizing was successful. Of course, depending on our business goals (such as for examples presenting certain ingredient, cooking tools or restaurant advertisements to our clients), we could decide to increase or decrease the number of user-target groups. However, as such a clear objective is not actually define, we may consider our first objective attained.

Cat. 4 users like the most :

Caramel Apple Pork Chops 4.4912170639899625

Asian Lettuce Wraps 4.665404040404041

Burgundy Pork Tenderloin 4.352697095435683

Balsamic Roasted Pork Loin 4.6142520612485285

Italian Breaded Pork Chops 4.519436997319035

Flatlander Chili 4.765174739423666

Firecracker Grilled Alaska Salmon 4.7916194790486974

Lamb Chops with Balsamic Reduction 4.816113744075829

Simple Turkey Chili 4.566982408660351

Spicy Bean Salsa 4.757142857142857

Marinated Pork Tenderloin 4.580908032596042

Lemon Garlic Tilapia 4.394828791055206

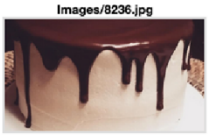
Mushroom Pork Chops 4.477434679334917

Fish Tacos 4.7348484848484835

Sage Pork Chops 4.4182590233545636

Cat. 3 users like most :

Satiny Chocolate Glaze 4.7762923351158655



No Bake Cookies III 4.5761843790012815

Basic Flaky Pie Crust 4.614088820826953

Ninety Minute Cinnamon Rolls 4.67406962785114

Soft Sugar Cookies IV 4.552321981424149

Chocolate Crinkles II 4.674914089347079

Cinnamon Rolls III 4.773782771535581

Pumpkin Bread IV 4.795957043588124

French Bread Rolls to Die For 4.59890610756609

Butter Flaky Pie Crust 4.6536312849162025

Cream Puffs 4.619302949061662

Caramel Popcorn 4.86036036036036

Cream Cheese Pound Cake III 4.6551373346897265

Simple White Cake 4.193340494092373

Buttery Soft Pretzels 4.5955555555555545

Cat. 8 users like most:

Lentil Soup 4.457703081232493

Black Bean and Salsa Soup 4.6771653543307075

Absolutely Fabulous Greek/House Dressing 4.633620689655173

Roasted Vegetables 4.644474034620505

Slow Cooker Taco Soup 4.635890767230169

Beef Stew VI 4.613746369796709

Sam's Famous Carrot Cake 4.783977110157368

Six Can Chicken Tortilla Soup 4.579842137219186

Roquefort Pear Salad 4.862688713156003

Chicken Milano 4.675980975029726

Chicken Tortilla Soup V 4.733585858585858

Asian Orange Chicken 4.453010279001468

Baked Slow Cooker Chicken 4.288546255506608

Playgroup Granola Bars 4.562318840579710

Italian Sausage Soup with Tortellini 4.8687537627934985

V. Building a simple collaborative filtering recommender system

In this last section we discuss our findings for the last task in this project : building an automated recipe recommender system for website's users.

Choosing the right type of recommender system

As our exploratory data analysis made clear, the website contains a few recipes that are very popular and many much less popular recipes. From this we infer that the website already offers recommendations to its users of the most popular recipes. Therefore It wouldn't add any value to the website if we built a recommender system that recommends recipes that are already popular. Instead, we need a recommender system that makes more personalized recommendation.

We can consider two options here : First, if the user is looking for a specific type of recipe, he may use keywords to search for it on the website. But even after clicking on a recipe that caught his attention, the user might actually want to try something else. For this purpose, it would be interesting to recommend a few other recipes to the user that are very similar in content to the recipe that is currently being browsed by the user : this comes down to building a content-based recommendation. We implement sucha content-based recommender system in he first paragraph of this section.

Frequent users of the website might also want to discover new recipes without having any specific idea in mind. For this purpose, it might be useful to build a collaborative filtering recommender system, which will be able to recommend new recipes to users based on the recipes they liked. This we will do in the second part of this section.

a/ Content-based recipe recommender system with TF-IDF

The content-based recommender system is extremely easy to build : for any given recipe, find k-nearest recipes using cosine similarity metric between each pair of Tf-IDF vectorized recipe, this time using not the list of ingredients but the complete list of instructions, so as to take other details such as cooking techniques into account. This simple model yield very satisfying results, as shown in the two figures below.

Examples of Tf-iDF content based recommendations

b/ K-Neighbors User-User Collaborative filtering recommender system

Lastly, I built a simple K-neighbors User-User collaborative filtering recommender system able to make more personalized recommendations. Such a recommendation system suffers from the cold-start issue. I therefore restricted this recommender system to users with at least ten recipe ratings.

The goal of a collaborative filtering recommender system is to predict the rating a given user is most likely to give to a given recipe that the user has not rated yet. Given this information, we can recommender to any user the recipes that the system has predicted he would give the highest ratings.

Why implement User-User collaborative filtering ? As we have seen during the Exploratory Data Analysis, there is not a great variability in the ratings users give to items : most users give almost only 4 or 5 stars to most of the recipes. What is important therefore in recommending recipes to users is not to recommend a recipe that the user is likely to rank high, which is most likely going to happen if the users actually choses to rate the recipe, but to recommend a recipe that is best suited to the user's taste.

Among the many possibilities one may chose from to build recommender systems, user-user collaborative filtering is one which is both relatively simple but at the same time is able to make very personalized recommendations. That is, instead of recommending the same items to all all users, the user-user collaborative filtering recommender system yields very different results for each user depending on which recipe the user has rated and liked most.

The reason for this is that the recommendations made for each user are based solely on users that have very similar tastes to the given user : in order to predict how user u will rate item i, we compute the weighted average of the ratings that users v most similar to user u gave to item i, the weights corresponding the cosine similarity between user u and user v. This way we are able to predict what rating any user u will give to any item i in a personalized manner.

In mathematical terms, the predicted rating by user u for a given item I is given by :

$$r_{u,i} = \text{aggr}_{u' \in U} \, r_{u',i}$$

where U denotes the set of top N users that are most similar to user u who rated item i.

$$r_{u,i} = \bar{r_u} + k \sum \text{simil}(u, u')(r_{u',i} - \bar{r_{u'}})$$

$$\text{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Evaluating the user-user collaborative filtering recommender system

One of the most common method for evaluating a recommender system is to compute how far of each predicted rating is from the actual rating (this is achieved by hiding some actual evaluations that are known to us during the recommender's training and comparing the recommender's predicted rating s to the known ratings).

As a baseline, we used a persistent recommender system : in this case each user-item predicted rating is equal to the average of the ratings all the users gave to that given item. With this baseline, we obtain a mean average error of **0.5613**.

Using the user-user collaborative filtering recommender method, we obtain a mean average error of **0.5607**, almost equal to the baseline. In other words, collaborative filtering achieves the exact same performance as the persistent model regarding the predictive ratings.

The difference, however, is that whereas the persistent model would recommend the same recipe to everyone, our collaborative filtering system offers very different recommendations to all users. Looking just at the recipe that the recommender system predicted the user would rate the highest (that is, the first recommendation), we see that most recipes were recommended only to a few people, a result which confirms that our recommender sytem is very personalized.

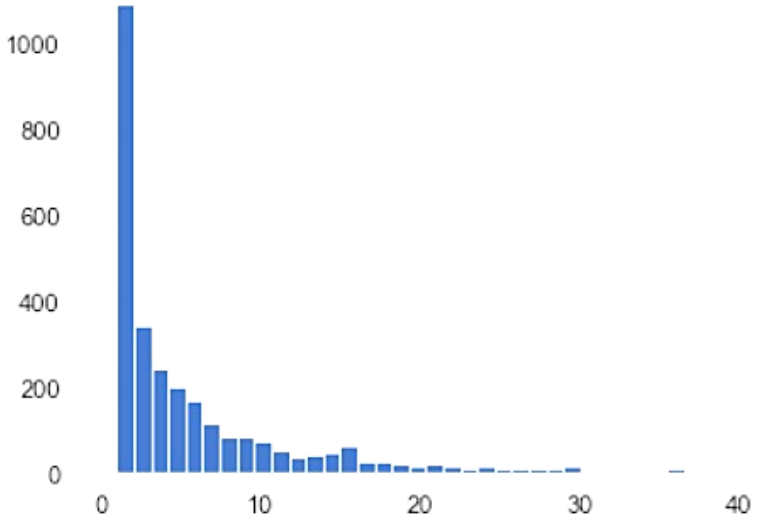


Figure 11. Distribution of the number of times a given recipe was recommended to users