A Culinary Tour of Data Science

Team 36

Maria Katergi

Davit Martirosyan

Carla Ohanesian

Iuliana Voinea

1 INTRODUCTION

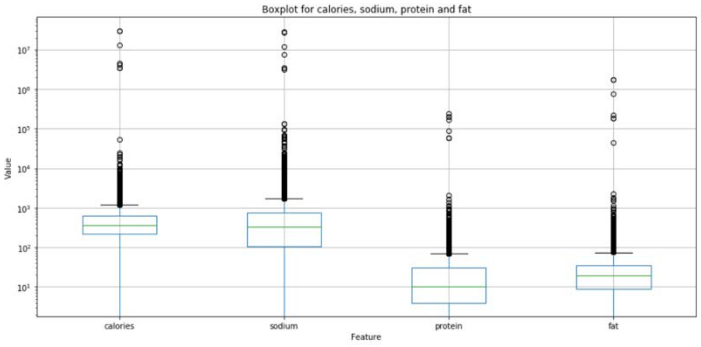
Ever since the moment people discovered fire, they have come up with various ways of combining and cooking different ingredients in order to obtain delicious recipes. Nowadays, the World Wide Web makes it possible for people to share and rate their favorite dishes on specialized platforms, while also specifying nutritional facts that are of importance for one's physical health and which might influence the final taste of the dish.

One such platform is Epicurious. In this project, we decided to explore how the recipes available on Epicurious are related to each other in terms of ingredients, categories and nutritional information. This can be done by creating a "recipe network" in the form of an undirected graph and analyzing its properties to draw meaningful conclusions. What made it possible to build such a graph is the fact that most recipes share ingredients, some more than others. Thus, it is interesting to see whether people prefer recipes that contain certain ingredients or that have certain ranges of nutritional factors. In addition, it is also highly interesting to see if a model can be built to accurately recommend new dishes based on a given list of input recipes.

In order to see all this, we used the Epicurious dataset provided on Kaggle[1] which contains more than 20.000 recipes with information such as the average rating, ingredients, nutritional aspects (calories, proteins, sodium, etc.) and much more.

2 GRAPH CONSTRUCTION

Given the nature of the raw dataset, some preprocessing steps had to be performed in order to be able to construct the graph. First of all, we discarded the 'desc' column containing the recipe descriptions, as it was not useful for our goal. Then, we removed the recipes with 0 rating -given that these corresponded to the recipes which were not rated on the Epicurious website- together with duplicates and recipes that had null entries in any of the columns. Moreover, after checking the ranges of the recipe features, we noticed some extreme values, for example a recipe had 60,000 calories, which is not possible. Therefore, we checked for outliers using boxplots (see the plot on the right). Based on the boxplots results and the findings about reasonable ranges for the nutritional factors studied here, we decided to



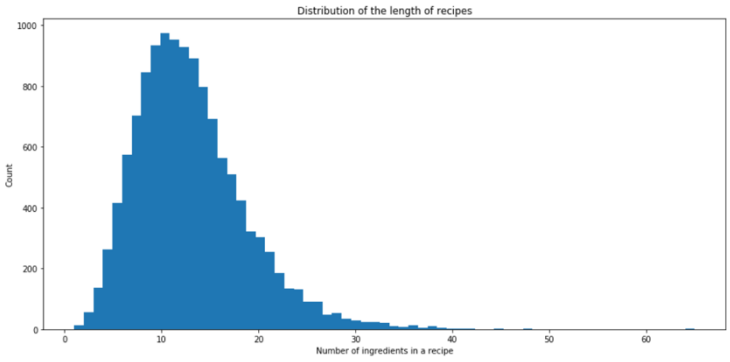
[1] https://www.kaggle.com/hugodarwood/epirecipes#full_format_recipes.json

threshold these values to remove the outliers. We were finally left with 12,466 recipes out of the initial 20,130.

Second of all, the most important step of the preprocessing consisted of building a list of ingredients for each recipe. The original 'ingredients' column contained them listed together with quantities, measures, stop words and utensils. As this was undesirable, we applied some Natural Language Processing steps in order to extract only the ingredients themselves. We used the NLTK Python library to do the following:

1. Tokenize and attach a part of speech tag to each word.
2. Keep only the nouns.
3. Remove unwanted words from a list we predefined containing quantities, measures, utensils, etc.
4. Perform lemmatization to map together different inflected and derived forms of the same ingredient.
5. Strip some characters which sometimes appeared appended to ingredients (e.g. *).
6. Keep only the unique words in the final ingredient list for each recipe.

Following these NLP steps, we ended up with reasonable lists of ingredients. After plotting the number of ingredients per recipe, we observed a slightly right-skewed Gaussian Distribution with most recipes having roughly 11 ingredients (see the plot shown on the right).



Building the cleaned lists of ingredients for every dish was crucial as we built our graph by connecting the recipes based on how similar they are in terms of ingredients. To compute this similarity, we used the Jaccard Similarity measure on the ingredients lists. Unfortunately, our data was still too large, thus we decided to subsample at random only 5000 out of the 12,466 nodes. This subsampling allowed us to use Gephi for visualization.

Finally, we constructed an adjacency matrix containing the Jaccard Similarity values as weights between every pair of nodes, representing the final graph made up of 8,685,453 edges and 5000 nodes. However, despite the large number of edges, most of the nodes were weakly connected. This also made our graph very dense. Hence, we inspected the Jaccard Similarity distribution and decided to only keep the connections with a weight greater or equal to 0.2. This way we reduced the number of edges to 230,352.

3 GRAPH STRUCTURE ANALYSIS

The following table summarizes the graph's properties:

Average degree	92.14
Sparsity	0.0184
Global clustering coefficient	0.3363
Average clustering coefficient	0.3152
Number of connected components	29
Percentage of nodes in largest component	99.44%
Diameter of largest component	7
Average shortest path length in largest component	2.7297
Degree distribution	Log-normal

4 COMMUNITY DETECTION – CLUSTERING

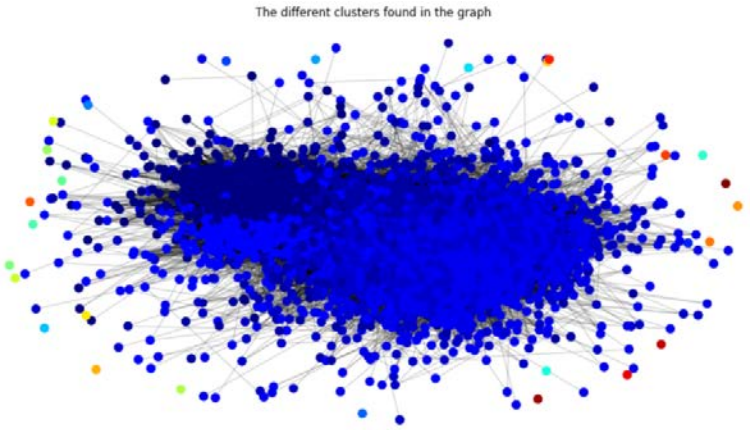
Given our graph is connected based on the similarity between dishes, it is interesting to identify the different clusters and investigate their different properties. To do so, the Louvain method for community detection was used. It works by optimizing the modularity, defined between -1 and 1, which measures the density of links inside communities compared to links between communities. The Louvain method is available in community in the NetworkX library.

Running the algorithm on our dataset, a total of 33 clusters were found. Among these clusters, 5 big clusters were identified, each having a different number of nodes: 1017, 499, 119, 1851 and 417 nodes. The other 28 clusters, each consists of one node. The 5 main clusters were investigated by looking at the dish titles and identifying the type of dish or cuisine of the recipes, and by extracting the top 10 categories for each cluster. The findings are summarized in the table below:

Cluster	Number of nodes	Example of dishes	Description
0	1017	Tarte Tatin, Pecan Shortbread Cookies, Apple-Raisin Bread Pudding …	Mainly comprised of deserts (e.g. cakes, chocolates, ice-cream, etc.). Some of the most occurring categories in this cluster include 'desert', 'bake' and 'chocolate'.
1	499	Mini Pizzas, Lettuce Soup, Mashed Potatoes with Kale, Potato Gratin with Goat Cheese and Garlic…	Contains mainly simple dishes which usually do not contain meat/chicken. 'Vegetarian' and 'Pescatarian' are for instance in the frequent categories' list. The recipes are mainly appetizers, sides and vegetarian simple dishes.
2	1188	Spanish-Style Shrimp and Scallop Salad, Beef and Avocado Fajitas, Nicaraguan-Style Steak…	Contains many Spanish and Latin American cuisine dishes. Some common ingredients are: avocado, bell pepper, jalapeño and beans. Also, most dishes are not vegetarian and contain meat (chicken, beef, fish).

3	1851	Melon and Prosciutto Risotto, Mussels alla Diavola, Duck Liver Pâté, Chinese Broccoli with Crabmeat, Coconut-Curry, Asian Noodle…	This cluster consists of a mixture of European cuisine (French, Italian...) and Asian cuisine. For instance, we can find gourmet French dishes and pasta, as well as noodles and curries.
4	417	Strawberries & Cream Frappé, Earl Grey Rum Punch, Strawberry-Banana Smoothie, Homemade Ginger Ale…	Contains mainly drinks, sorbets and ice creams. Relevant categories found in this cluster are 'Alcoholic', 'drink' or 'Cocktail Party', which reflect very well the recipes found in this cluster.

The visualization of the graph with networkX showing the different clusters is presented below:



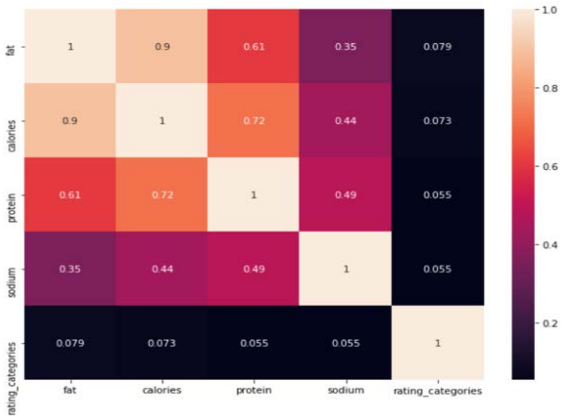
5 LOGISTIC REGRESSION AND GRAPH SIGNALS

One interesting question that one might think of is the following: What are nutritional factors that affect people's culinary preferences? In order to answer this question, we thought of training a logistic regression model, having as input dishes together with their nutritional values, namely the amount of calories, protein, fat and sodium. In the case where this model was able to correctly predict the rating of a dish, based on its nutritional aspects, therefore we could have concluded that the nutritional information of a certain dish influences people's culinary preferences and hence their ratings. However, we noticed that the accuracies that our model showed were not promising.

We then extracted some features using graph filters by denoising the available nutritional factors features. Again, the results did not show much improvement.

The first and main reason for this is the data itself; we categorized the dishes based on their ratings, with labels {0,1,2} respectively for bad (< 3), medium (3< x < 4) and high ratings (>4). However, we noticed that there was an important imbalance in the categories as most dishes had high ratings and hence most data points belong to class 2 with very few points belonging to

class 0 (less than 5%). This explains why the model predicts high ratings to all the dishes in the test set. On the other hand, looking at the correlation matrix shown on the right, we can clearly see that some of our predictors are quite correlated with each other (e.g. 'fat' and 'calories' have correlation coefficient of 0.9), however, unfortunately, our predictors have almost 0 correlation with the target, namely 'rating_categories', variable. This is of course undesirable and is a sign that we cannot expect to get good predictive model. Thus, we concluded that nutritional aspects in dishes do not really influence people's culinary preferences and hence their ratings.



6 KNN RECOMMENDER SYSTEM

In general, one of the key factors affecting business growth and success is how personalized they are able to make their products for their users. More is the personalization, more is the user engagement and satisfaction. This said, we were eager to build a simple recommender system that could wisely utilize our graph and produce proper recommendations for a given list of recipes. To do this, we implemented a KNN based recommender system algorithm which uses the following steps to produce recommendations:

1. Take the k-highest rated recipes, K, from the given list of recipes
2. Iterate through all k chosen recipes giving a score to each connected recipe i as follows

$$score_i = \sum_{k \in K} jaccard\_sim(i, k)$$

3. Sort the recipes based on the scores in descending order and return the first n recipes (excluding the ones appearing in the input list)

This is a simple, yet a powerful algorithm capable of making adequate recommendations. An example illustrating how the algorithm worked for our case is presented below:

For the input list presented in Appendix I, our system (for k=10) outputs the following top-5 recommendations:

- Butter Pie Crust Dough
- Classic Sour Cherry Pie with Lattice Crust
- Florida Punch
- Peaches in Ginger Syrup
- Cantaloupe Granita

As one can observe, the recommendations are quite appropriate given that the input list consists of deserts and drinks.

I. APPENDIX I

This list consists of deserts (e.g. cake, cookie) and drinks (e.g. lemonade, smoothie).

1. 'Crushed-Mint Lemonade',
2. 'Ginger-Honey Lemonade',
3. 'Papaya Smoothie',
4. 'Oranges with Pomegranate Molasses and Honey',
5. 'Pine Nut Brittle',
6. 'Sweet Avocado Mousse',
7. 'Gimlet',
8. 'Campari Citrus Cooler',
9. 'Tangerine Granita',
10. 'Chocolate, Cherry and Marsala Cassata',
11. 'Ginger Pudding',
12. 'Trifle with Strawberries and Caramel-Coated Bananas',
13. 'Thin Apple Tarts',
14. 'Shortbread Cookies',
15. 'Burnt-Caramel Custards',
16. 'Peach White-Wine Sangria'

II. APPENDIX II

Graph visualization using Gephi with MultiGravity ForceAtlas 2 layout is found below: