# Ingredients For A Good Recipe

**Marco Bearzi**
marco.bearzi@epfl.ch

**Jacopo Leo**
jacopo.lea@epfl.ch

**Elias Gajo**
elias.gajo@epfl.ch

**Maxime Lautenbacher**
maxime.lautenbacher@epfl.ch

## Abstract

Food has always been a topic that peaks the interest of people. Now more than ever, recipes can be found all over the internet, and are used all over the world. As the rating of the recipes are often put forward, this is usually the only element people base their choice on. But what ingredients make the recipe get a good rating ?
The aim of this project is to find correlations between the ingredients that make up a recipe and that recipe's rating.
The recipes and their ratings are gathered from different popular online recipe websites. The ingredients are scraped from the websites, and using different text extraction methods as well as online quantity and density normalization tables, the weight and percentage of each ingredient of the recipe is obtained.
After having acquired this data, different machine learning algorithms are performed on the dataset obtained to create a model that can predict whether a recipe will have a good rating or not.

## 1 Introduction

The aim of this project is to determine if it is possible to find a correlation between the ingredients that make up a cooking recipe and its given rating. To carry out this project idea, the ”Cooking recipes” dataset is used. The first focus of this project is to be able to extract the rating and all the ingredients and their quantity from the acquired recipes. This data is then normalized to be able to use different adequate machine learning methods. Combining the different models will provide a final model that will predict if a recipe is likely to get a good rating based on its ingredients.

## 2 Dataset Description

The dataset used for this project is ”Cooking recipes”. It contains cooking recipes from various websites. The recipes are all HTML documents, but as they are from different websites, the HTML page architecture also changes. Not all the documents are recipes, and not every recipe has a rating, so these pages are useless in the scope of this project.

## 3 Data acquisition

The HTML documents are parsed using BeautifulSoup to retrieve the information useful to the project. The parsing is a tedious process that has to be done on a website domain to website domain basis. As there are a lot of different website domains, the first task was to identify which ones had the most recipes. The most popular websites are shown in Figure. 2:

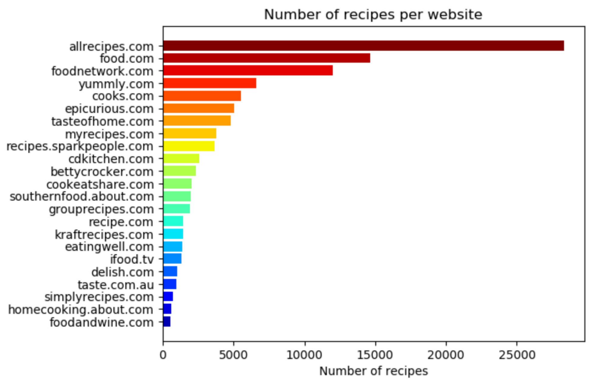


Figure 1: Number of recipes for the top websites

At the sight of this plot it is clear that it is meaningless to try and parse through every websites. The three most popular websites contain 50% of all the recipes, and they provide a sufficient quantity of recipes to perform the data analysis. However, the websites have to have a ”rating” section to be usable, as it is the essential element of our analysis. Luckily, all the top three websites have a rating section. So a parsing code was written for each of the following top three websites: *allrecipes.com* with , *food.com* and *foodnetwork.com*

## 4 Feature extraction

The recipes contain different features that get extracted during the parsing. The extracted features are shown in the following list:

- Website name
- Recipe name
- Preparation time
- Rating
- Number of reviews
- Ingredients
- Quantities
- Units

All those features are presented in different scales, units and can even be missing. They have to be converted into a common format in order to be enable us to perform a valid analysis. The dataset is about 12 GB large, which makes the parsing possible on a single computer, although it takes some time. The complete parsing took around 3 hours, and the results were saved in a pickle file in order not to have to parse them every time.

Although all these features were extracted, the final models only take into account the rating and the ingredients of the recipes.

## 5 Data cleaning

The data had to be cleaned and normalized in order to be usable. The preparation times that were given in hours or days were converted to minutes. The ratings and number of reviews had to sometimes be converted from strings to numerical values.

However, the main part of the data cleaning was to be able to recover only the actual ingredient from the string found on the website. There are a lot of ingredient strings that contain futile words not specific to the ingredient itself. To do this, the unique ingredients across the recipes were observed, and a list of the futile words was constructed by hand.

The following plot shows the number of times the top 25 ingredients are seen throughout all the recipes:

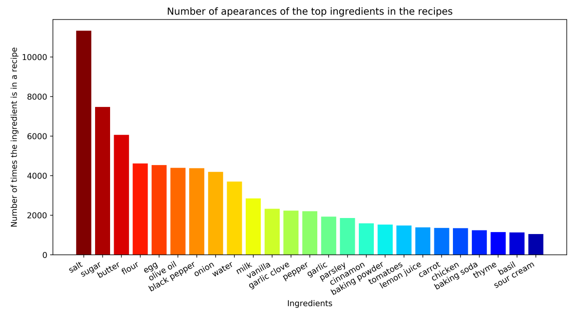


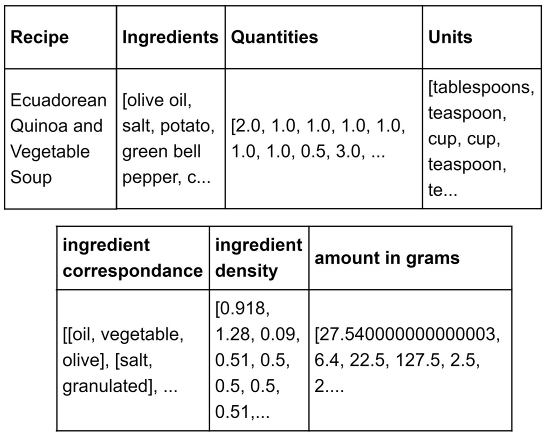
Figure 2: Number of times the top ingredients are seen in the recipes

## 6 Ingredient quantity computation

At first, the aim was to only extract the ingredient's name without the quantity, and just consider if it is present in the recipe or not. The models were first ran with this simple assignation. However, at the end of the project, it was decided to also try and extract the quantity of every ingredient in order to have the percentage they take up in the recipe. This allows to know how the ingredients are distributed in the recipe, instead of considering that they all have the same importance.

However, not all the ingredients have the quantity written next to them in kilograms for example. A lot of them have different norms before them, "cups", "spoons", "steak", ... To convert these terms to kilograms, normalization tables found on the internet are used (1) (2). One of them contain the usual weights for different of these common terms such as "steak" or "wings". Another one contains the density of liquids used in cooking. This table enables the conversion from "cups" or milliliters to kilograms.

To illustrate what the program does, the example table in Figure. 3 will be explained. The table contains first the name of the recipe and the ingredients that composes it. Then, the column "Units" shows the unit of the ingredient found in the ingredient string. Similarly, the quantity (of units) of the specific ingredient is stored in the column "Quantities". Then, the columns "Ingredient correspondence" is filled by comparing the ingredients of the recipe with the ingredient strings in a document containing the densities of common ingredients (1). Once the correspondence between the ingredient and the table is found, the "Ingredient density" column can simply be filled by looking at the density value in the document (1). A function to convert the volume quantities such as "tablespoon" was created in order to have all the liquid ingredients in milliliters for the final computation. This was done using the volume correspondences found online (2). Finally, the column "Amount in grams" is filled by multiplying the quantity with the with the unit and corresponding density. If the ingredient was already given in grams, it is kept that way.



| Recipe | Ingredients | Quantities | Units |
|---|---|---|---|
| Ecuadorean Quinoa and Vegetable Soup | [olive oil, salt, potato, green bell pepper, c... | [2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 3.0, ... | [tablespoons, teaspoon, cup, cup, teaspoon, te... |

| ingredient correspondance | ingredient density | amount in grams |
|---|---|---|
| [[oil, vegetable, olive], [salt, granulated], ... | [0.918, 1.28, 0.09, 0.51, 0.5, 0.5, 0.5, 0.51,... | [27.540000000000003, 6.4, 22.5, 127.5, 2.5, 2.... |

Figure 3: Table with the recipe's ingredients, their quantity, units, correspondence in tables, density and amount in grams

When all the ingredients are converted to grams, the percentage of each ingredient in the recipe can be calculated.

## 7 Feature selection

To compute the machine learning models, only a selection of the ingredients are kept, in order not to have too many variables and not to overfit the model. It was decided to keep only the ingredients that appear at minimum 200 times throughout all the parse recipes. Then, to narrow it down more, the correlation matrix of those is observed. Only the ingredients with a correlation higher than 0.01 in absolute value are kept, the other ones are discarded as they don't carry a lot of information for our model.

The Figure. 4 shows the correlation of the many ingredients with the rating. The dotted red lines show the separation between the ingredients taken into consideration and those discarded for the models. The higher correlations are kept to get a better model.

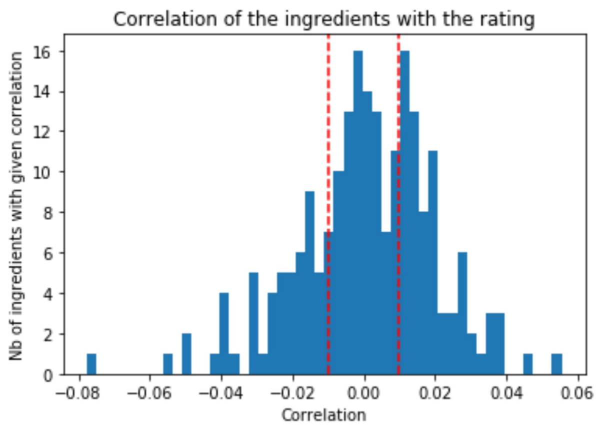


Figure 4: Correlation of the ingredients with the rating (for the binary dataframe)

## 8 Machine learning models

Two datasets were trained and tested. The first one is a binary dataset of our recipe, just taking into account if an ingredient is present in the recipe or not. The second one takes into account the quantity of each ingredient. As explained previously, the quantity of each ingredient is calculated and stored in the dataset in grams.

### 8.1 Ridge Regression model

The first machine learning model that is created has to be able to predict the rating of a recipe. To do this a ridge regression model was trained. To optimize our model and reduce the error, the best hyperparameter "$\alpha$" was computed by testing different values and performing cross validation. Different optimal "$\alpha$" parameters were found for the quantity and binary dataset as seen in Figure. 5:

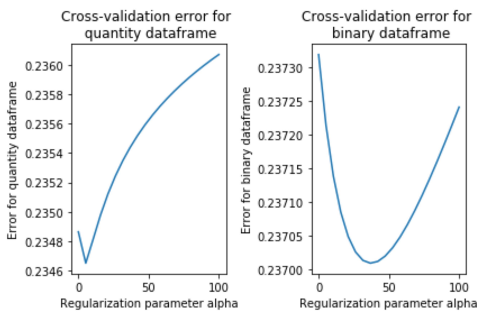


Figure 5: $\alpha$ parameters for both dataframes

The errors obtained for both dataframes are shown in the Table 1 below:

| | |
|---|---|
| Quantity | 0.241 |
| Binary | 0.237 |

Table 1: Errors of both dataframes for the ridge regression model

It is seen that the errors are very close for both datasets. This means that unfortunately scraping the quantities of each ingredient doesn't improve the model.

### 8.2 Classification models

The next idea was to use classification. To do this the recipes were separated into "good" and "bad" ratings. The recipes above 4.5 were considered good, and the ones below considered bad. The aim was to determine in which category a recipe is.

#### 8.2.1 Logistic Regression model

The first classification model implemented was logistic regression. Different values of threshold to separate the good and bad recipes were tested, and 4.5 gave the best results. So the final model was trained with a threshold of 4.5. After the model is trained and tested, the following confusion matrix is obtained:

| | | Class | |
|---|---|---|---|
| | | True | False |
| Classified | True | 1496 | 1153 |
| | False | 2840 | 1819 |

Table 2: Logistic regression confusion matrix with the binary dataframe

Table. 2 shows clearly that the model classifies more often correctly than incorrectly. However, the model is far from perfect because there are still a lot of misclassifications. The F1-score obtained for this model is 0.579.

#### 8.2.2 Random Forests

The next model tested is random forests. The model was trained with several forest depths to find the smallest error. The final depth used is 12. The confusion matrix is shown in the following table:

| | | Class | |
|---|---|---|---|
| | | True | False |
| Classified | True | 2163 | 1686 |
| | False | 1640 | 1819 |

Table 3: Random forests confusion matrix with the binary dataframe

The table 3 shows that again it classifies more often correctly than incorrectly. For this model the F1-score is 0.579.

#### 8.2.3 KNN

The next model tested is KNN. The model was trained with several neighbors to find which parameter gives the best result. The number of neighbors used for the final model is 34. The confusion matrix is shown in the following table:

The F1-score is for this model is 0.55. It is a bit lower than the two previous models.

| | | Class | |
|---|---|---|---|
| | | True | False |
| Classified | True | 2131 | 1601 |
| | False | 1658 | 1918 |

Table 4: KNN confusion matrix with the binary dataframe

### 8.2.4 Neural Networks

The final model trained was neural networks. It gave the best result with 10 layers. The confusion matrix is showed in the following table:

| | | Class | |
|---|---|---|---|
| | | True | False |
| Classified | True | 1660 | 1798 |
| | False | 1305 | 2545 |

Table 5: Neural Networks confusion matrix with the binary dataframe

Table. 5 shows another similar classification. The F1-score is 0.569 for this model.

### 8.2.5 Combining three models

The last model was trained by combining the 3 different models (Random Forest, KNN and Neural Networks) using the best parameter for each model. The models are trained over the same train set and then tested on the same set. The final prediction of weather a recipe is good or bad is done by taking the majority result of the three models. The confusion matrix of this combination is shown in the following table:

| | | Class | |
|---|---|---|---|
| | | True | False |
| Classified | True | 1691 | 1799 |
| | False | 1331 | 2487 |

Table 6: Neural Networks confusion matrix with the binary dataframe

Table. 6 shows that the new prediction model is unfortunately not much better than the three separate ones. The F1-score of this combination is 0.56 which is not as good as logistic regression or random forests alone.

## 9 Conclusion

Unfortunately, the error was too big to come up with a model that could correctly predict the rating that a certain recipe would have. For the second part using classification, the models can come up with an F1-score higher than 0.5, which shows that it is possible to a certain extent to predict if a recipe is going to be ”good” or ”bad”. Unfortunately the models are not great, and quite a lot of misclassifications are obtained. This suggests that it is not possible to create a model that can precisely predict the rating of recipes based on their ingredients with this dataset. This isn’t very surprising, because the rating is probably quite random. Indeed, different people will follow the recipes differently and this can affect the final product and thus the rating. Additionally, people have different tastes and will perceive meals various ways. Finally it is also probable that the ingredients don’t have a huge impact on the rating, as even a commonly liked ingredient can be wrongly used in a recipe and create a dull meal.

There are a lot of points that could explain these bad results :

- We try to predict the rating of the recipes by it’s ingredients. But our dataset is very general, in the sense that it contains a very large variety of dishes such as desserts, soups, salted dishes, meat, ... We would have probably obtained better results if we considered some specific categories of dishes before trying the prediction models. Indeed the sugar effect would certainly be different if the dish is a dessert or a soup.

- We obtained better results with the binary dataframe than with the quantity dataframe, which is not intuitive. However, we can explain this result by observing the dataset. The quantity of the ingredients is crucial for the result of a mixture, but is a recipe a mixture ? Not necessarily... For example, a recipe of a steak and some rice isn’t a mixture. We can for instance double the quantity of rice without impacting the success of the recipe. So an improvement of our project would be to focus only on the recipes which correspond to a mixture of ingredients.

- Another issue will be to think about the creation of the dataset itself. The dataset consists of recipes that have been written mainly by cooks, which means that these recipes should already be good. A proof of that is that most of the recipes have a rating higher than 4, and the dataset lacks of bad recipes. Hence, this dataset is very useful for people who search good recipes, but not really useful for us. Indeed, we want to make our program learn if a recipe is good or bad. Then we need a dataset composed of a huge number of mixture of ingredients with their rating, but not recipes, as a recipe is already an ”optimal” mixture. Then, the answer to our bad result is the same as the answer of the following question : How can the model learn what is a good mixture if it doesn’t know what is a bad mixture ?

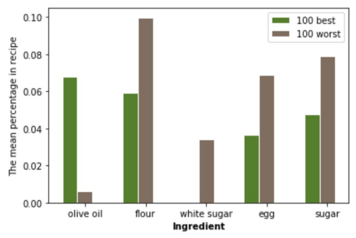


Figure 6: Percentage of ingredients in the 100 worst and best recipes

- The plot in Fig. 6 above shows how the mean percentage varies between the good and the bad recipes. The plot contains only the 5 ingredients on which the difference is the biggest. Even in this plot, the sugar doesn’t seem to identify well a good or a bad recipe. And all the other ingredients which are not present in this plot differentiate less the good and bad recipes. Hence, basing our prediction on all our 200 ingredients can’t give good result. The only good features that seem to be good candidates for the prediction are the olive oil and the white sugar, as we observe a clear difference for them between good and bad recipes.

## References

(1) http://www.fao.org/3/ap815e/ap815e.pdf
Density conversion table [Accessed 19.12.2019]

(2) https://en.wikipedia.org/wiki/Cooking_weights_
and_measures
Volume conversion table [Accessed 19.12.2019]