David Chang and Jared von Halle

CSC 478

Final Project

Due 6/6/16

# Introduction

We chose to perform data analysis on the World Food Facts data provided by the Open Food Facts Database. The World Food Facts dataset comes in the form of a single table, where each row is a different food. The columns represent facts about each food. Facts include textual information such as what countries the food is sold in, the generic name of the food, the product name, information about packaging, the branding, nutrition labels, and much more. There are also many columns related to the nutriment contents of each food. This data is given in the form of how many grams of a nutriment the food contains per 100 grams of the food. All in all, there were 159 features that describe 1 particular food.

# Objectives

Our objective, in the most general sense, was to explore the World Food Facts data to determine if we could find patterns between different foods or different qualities of foods. We were particularly interested in exploring what we could predict based on the nutrition and nutriment makeup of foods.

One angle we took was to see if we could find patterns between the nutriment makeup of foods and the countries that they are sold in. We hypothesized that different countries would have tendencies to sell foods with different nutrient makeups. Therefore, we were hopeful that we could predict what country a food was sold in based on the contents of that food. We used various classification methods to attempt this, which are described further in the sections below.

We also sought to predict the nutrition score of a food based on the nutriments that it is made up of. The nutrition score, which was included in the data for each food, is a calculation of the food’s overall health from the UK Food Standards Administration. We used regression analysis techniques for this prediction, discussed more in the sections below.

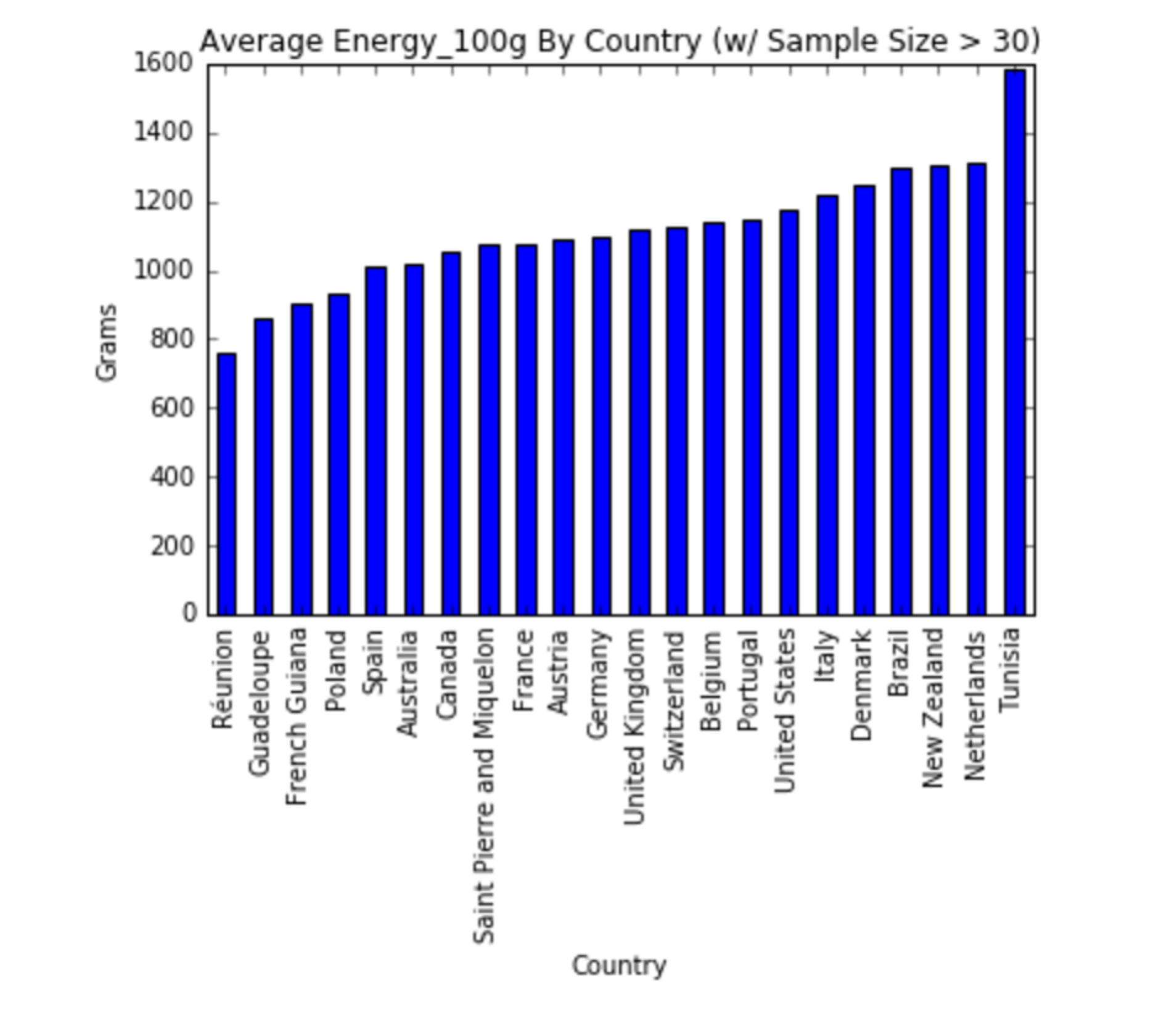
Finally, we sought to separate foods into various buckets based on the their nutriment makeup. The data included a “category” field, where each individual food is put into a subjective category of food such as meat, dairy, sugary snack, etc. We explored techniques for classifying each food into these categories based on their nutriment makeups. We used clustering techniques to accomplish this task.

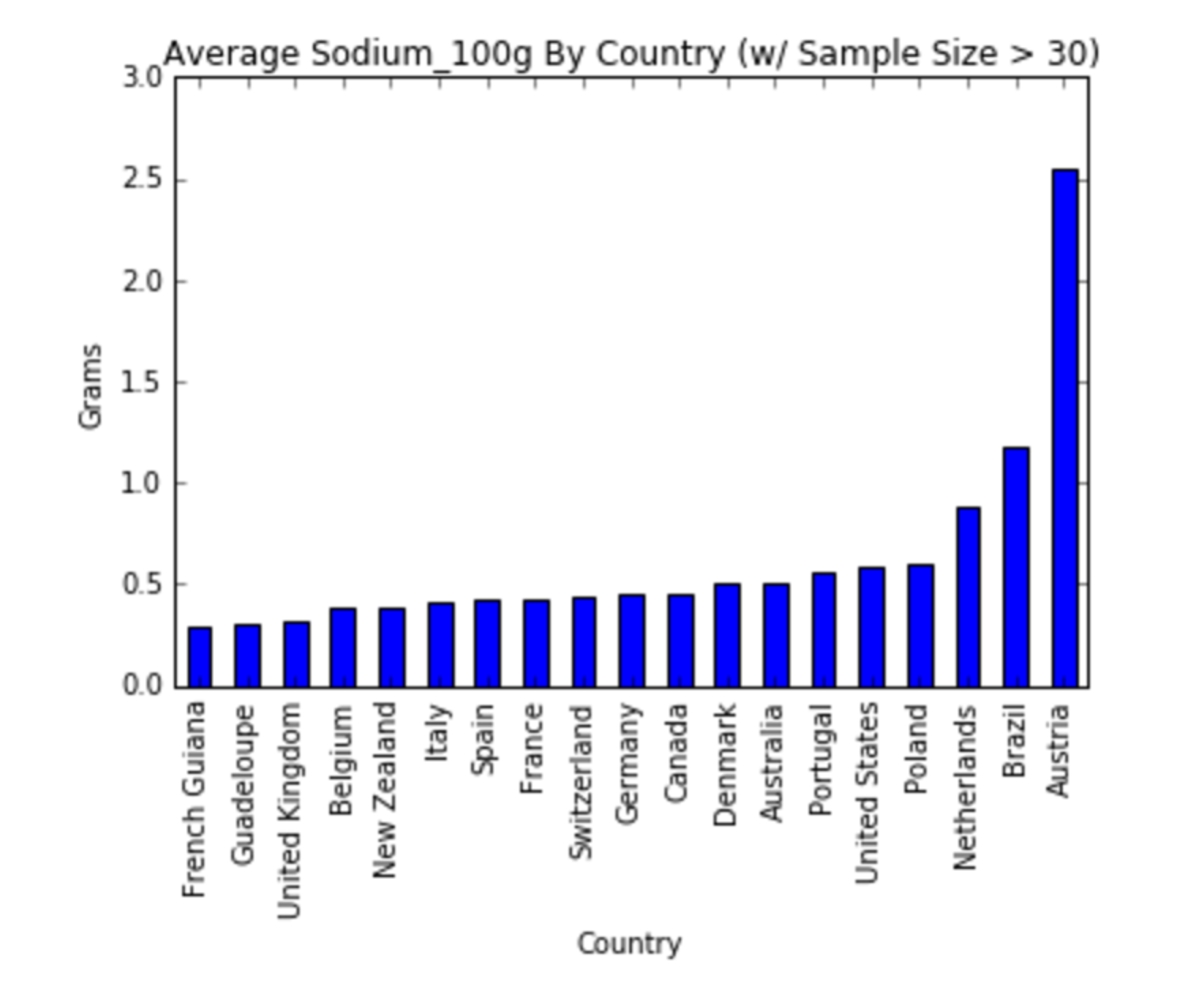
# Data Exploration (Data\_Exploration.ipynb)

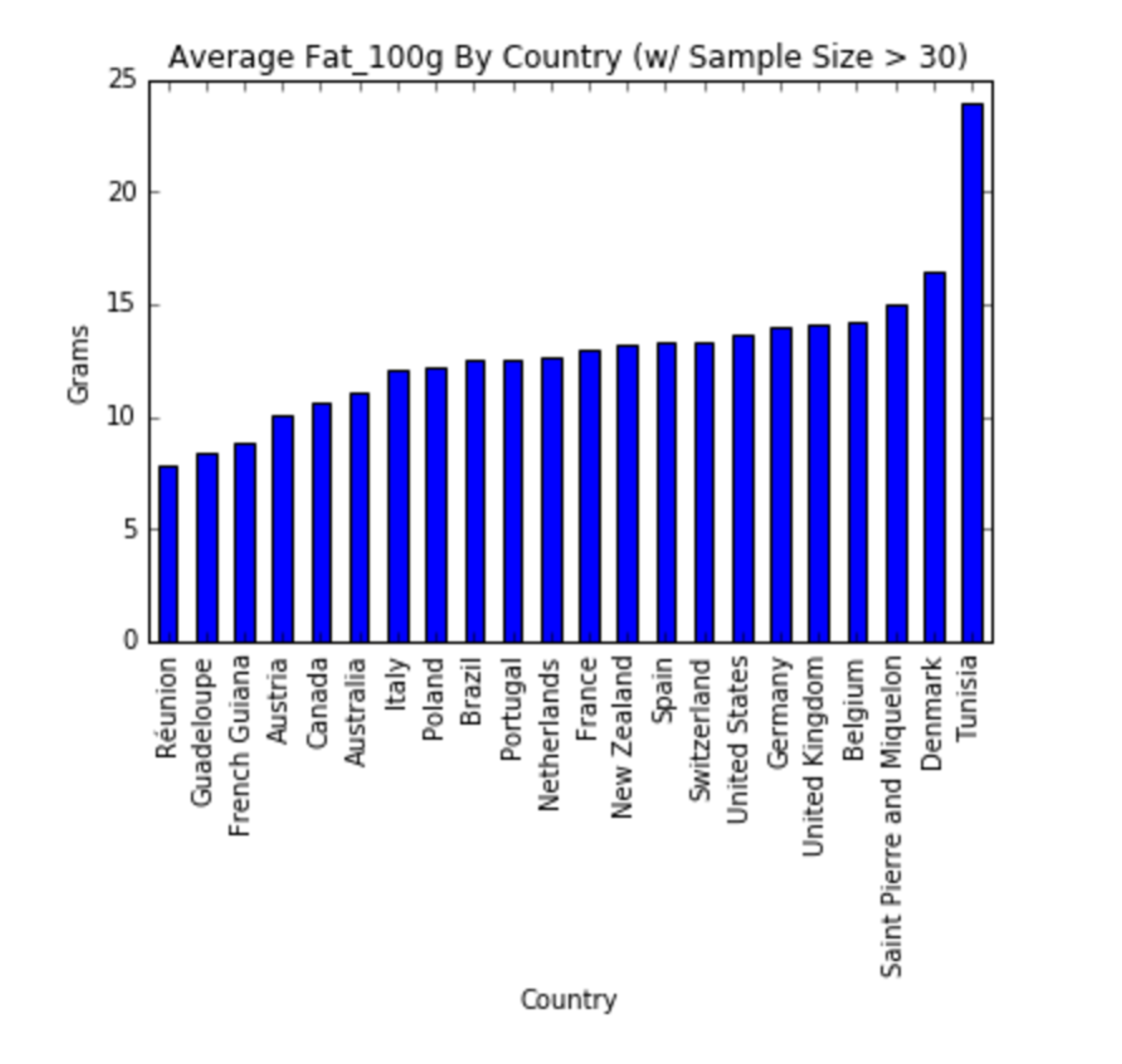
In order to get a general feel for the dataset that we chose, we decided to look at the differences between countries in terms of various nutriment features. The first thing we noticed about the data set was that it was very sparse. A large portion of the foods had no nutriment information at all so we knew from the get go that the sparseness of the data could be a major obstacle for us.

The first thing we did to get used to working with the data was to split the information based on countries in which the food is sold. There is a feature called "countries\_en" that lists all the countries in which the food is sold. We examined only a subset of the unique country values. For each nutriment value, we chose the subset of countries that had at least 30 foods where the nutriment value was not Null.

We created a python function in the "utilities.py" file. This method is called "compareCountriesByNutrimentAverage" and it takes in the full dataset, the unique country names, and the nutriment we want to compare each country by. The function outputs a pandas dataframe that contains the chosen nutriment averages for all the countries that we chose. Below are the plots that we generated. The nutriments that we chose were: sugars, energy, sodium, fat, proteins, and carbohydrates. Here are a few examples:







We were able to glean some interesting observations from the data by doing this.

* Austria has by and far the most sodium per 100 grams of food.
* Tunisia has significantly more fat per 100 grams than any of the other countries and thus also had the most energy per 100 grams of food.
* The United States was never the worst country for any of the categories we chose but consistently fell in the top 25%.

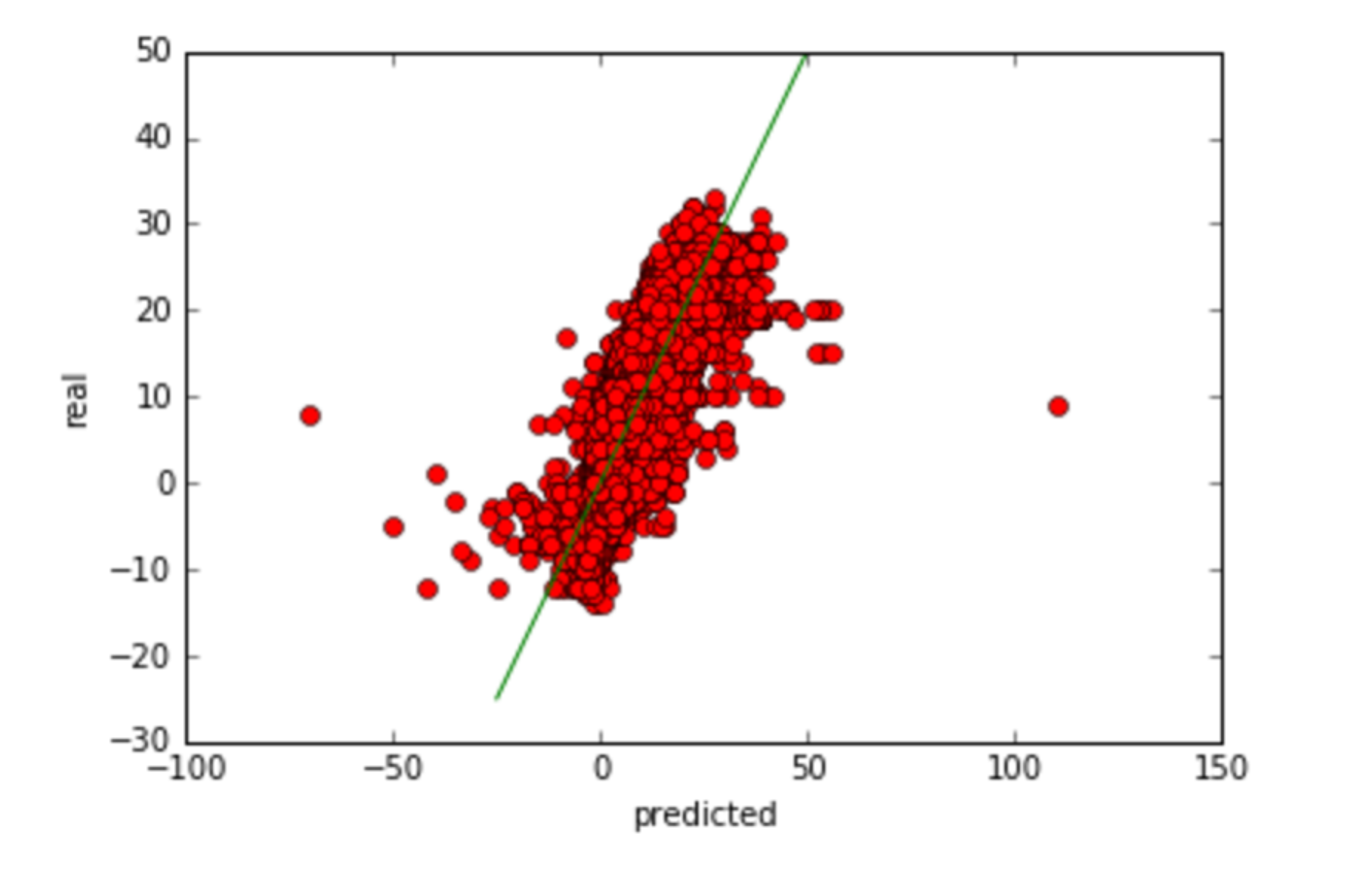
Also, out of curiosity we looked at what the most sugary food listed for the US was and it turned out to be a lemon candy called "Super Lemon"... who knew?

# Regression Analysis (Regression.ipynb)

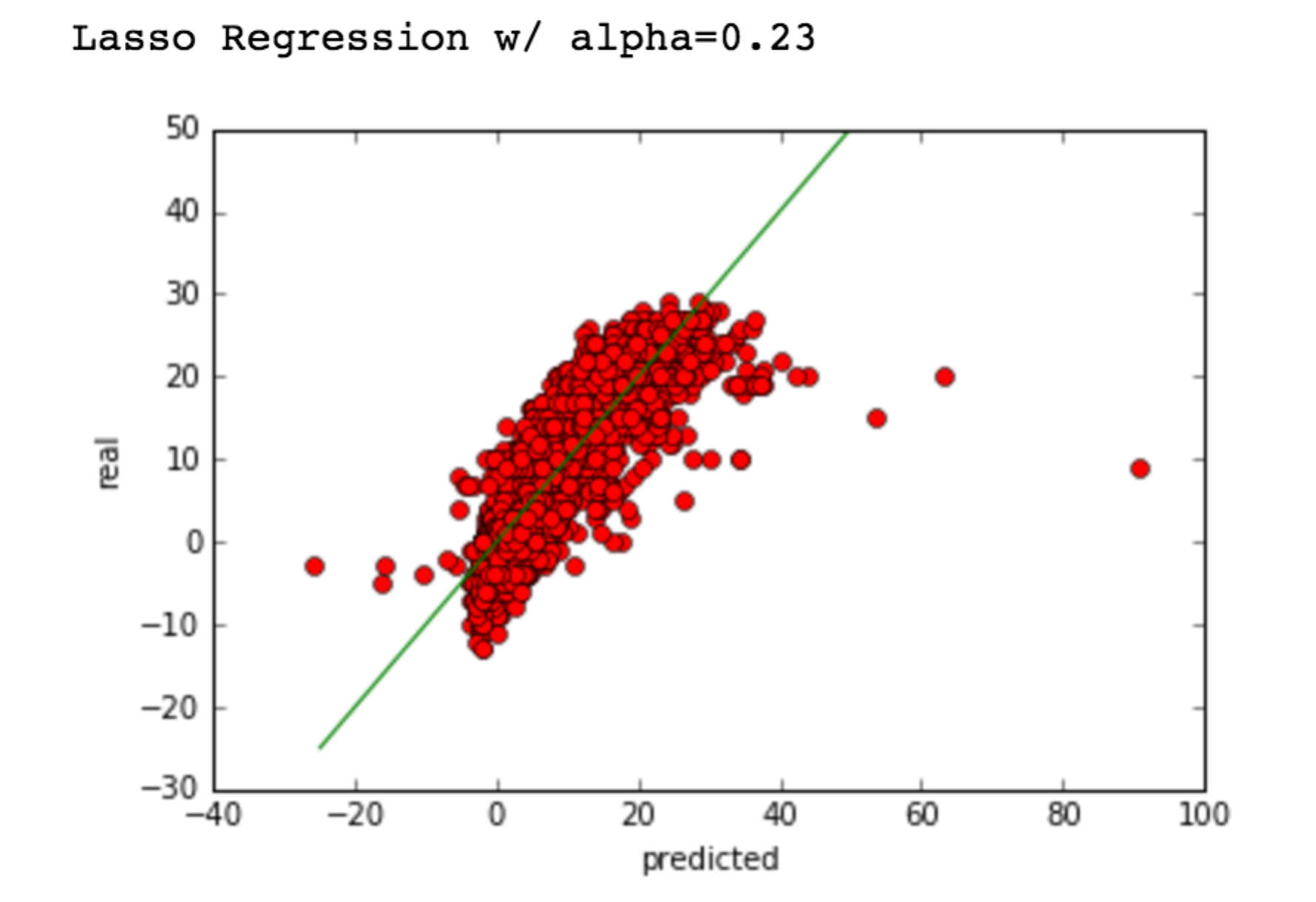
The data contains a particular feature called "nutrition\_score\_uk\_100g". This field contains a numerical score determined by the UK Food Standards Administration (FSA). This led us to ask the question "Can we predict the nutrition score based on the nutriments listed for the food"? Because this score was numeric, regression analysis was used.

We limited the data to all the nutriment values, which are also all numeric. The target dataset was the nutrition score. We then filtered out all the rows in the dataset that had no nutrition score because those rows would not be helpful for building and testing a regression model. Any nutriment value that was still null was filled with 0 to signify that there is none of that particular nutriment in the food.

The first thing we tried to do was use linear regression to predict the score. After performing linear regression on the training set we attained an root mean squared error of only 4.6161 and after plotting the regression line, we saw that there was a great fit. A RMSE of 4.6 felt good because the scores ranged anywhere from 37 to -14. However, when we applied linear regression using 10-fold cross validation, our RMSE skyrocketed to 1418.3! This led us to conclude that simple linear regression was not the tool needed for the job.



The next thing we did was try 3 other linear regression models in parallel to find the best linear regression and alpha combination. We used lasso, ridge and elastic-net to try to get a better model than linear regression. We believed that the reason linear regression was not good was because the data was pretty sparse and we were over-fitting to training data. Ridge, Lasso and Elastic-Net regression models are better at handling sparse datasets. We calculated RMSE on the dataset with the 3 different regression models and 50 different alpha values between 0.01 and 0.5. The output can be seen on the attached ipynb file for regression.

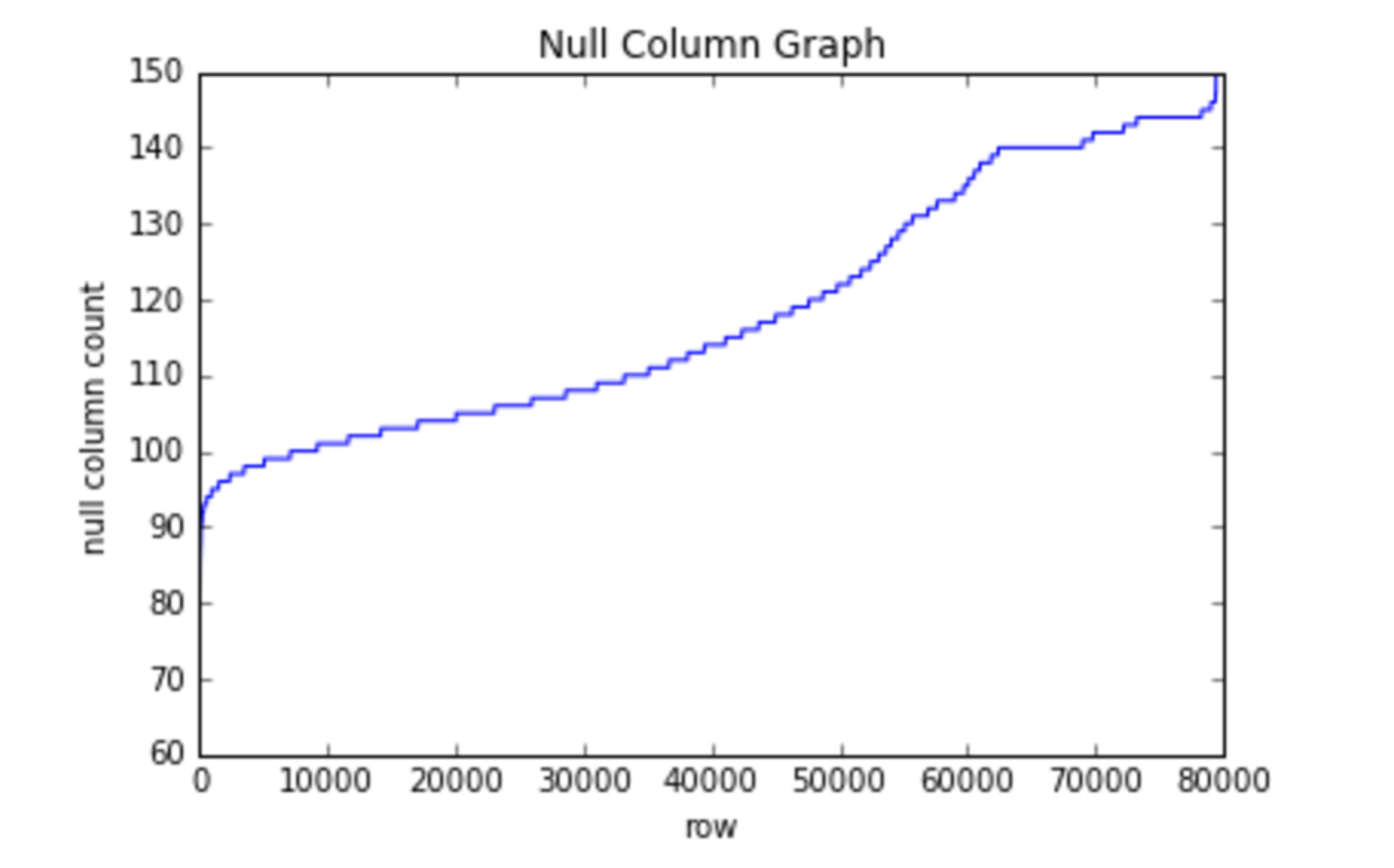


The best regression model/alpha value turned out to be Lasso regression with an alpha value of 0.23. The RMSE turned out to be 4.709 for the chosen combination. Lasso regression was highly successful at predicting the nutrition score.

# Predicting Where a Food is Sold (Classification.ipynb)

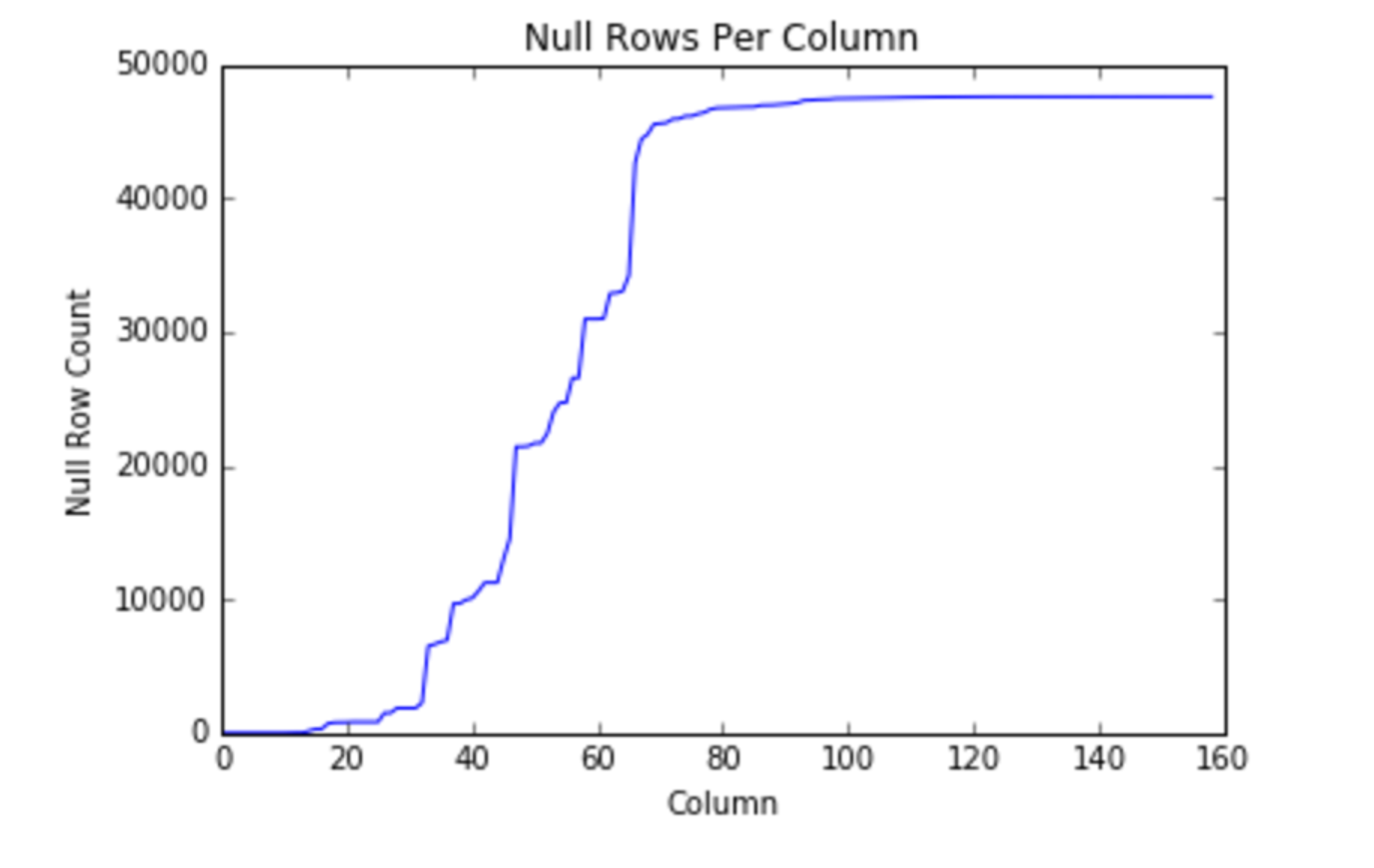
## General Data Preprocessing

The World Food Facts data is very sparse. We did not want (nor could we) completely ignore all sparse fields in the dataset. However, we for this section, we did remove rows and columns that were excessively sparse. We first examined the number of columns that were null for each row. The plot below displays the number of null fields for each row in the data.



There were 159 columns in total in the graph. You can see in the graph above that for some rows, almost all of the fields were null. For this section, we decided to remove any rows that had 120 or more null fields.

We then examined the sparseness of certain fields (columns) in the data. The graph below shows the number of rows that contain a null value for each column.



After removing the sparse rows in the previous step, there were 47,586 rows left in the data. In this case, we decided to remove columns from the dataset where more than 35,000 rows contained a null value for that field.

This trimming left us with a matrix of size 47,586 x 66, which is down from an original dataset of size 79,470 x 159.

For the sake of predicting what country each food was sold in, we did not want to consider foods that were listed as being sold in multiple countries. These foods represented a very small portion of the data (2.8%), so we removed these rows. This left us with a matrix of size 46,210 x 66.

Since our goal was to predict the country a food is sold in based only on its nutriment makeup, we limited the records for this section to include only the columns for nutriment values. This reduced the size of the matrix to 46,210 x 11.

We replaced all null values with the mean from each column and normalized the data. The final pre-processing step was the split the data into training and test sets.

## Nearest Centroid Classifier

The first technique that we attempted for the country classification was Nearest Centroid Classification. This is similar to the Rocchio method that is used for text classification, in that a prototype vector is calculated for each category. In our case, the categories are the countries that Foods are sold in. Our technique differed from Rocchio, however, because we were not dealing with text classification. Therefore, it didn’t make sense for us to use the TFIDF approach. Instead, we calculated the prototype vectors by calculating the average of each vector in the training set pertaining to each category (each country). We accomplished this by adding the vectors in the training set together for each country, and then dividing by the number of vectors that were added together.

Our main motivation for utilizing the Nearest Centroid Classification approach was performance. With this large dataset, we expected a performance gain from this approach because much of the work can be accomplished prior to the actual classification of an individual row. Specifically, we could calculate the prototype vectors using the training set ahead of time. During the actual classification, the row being classified only needed to be compared to the prototype vectors. This is opposed to other approaches where the row being classified must be compared to all other rows in the training set during the actual classification step.

After creating the prototype vectors, we used the test portion of the data to see how well the approach worked. We used cosine similarity as our distance measure, looking for the prototype vector that was closest in distance to each row in the test set.

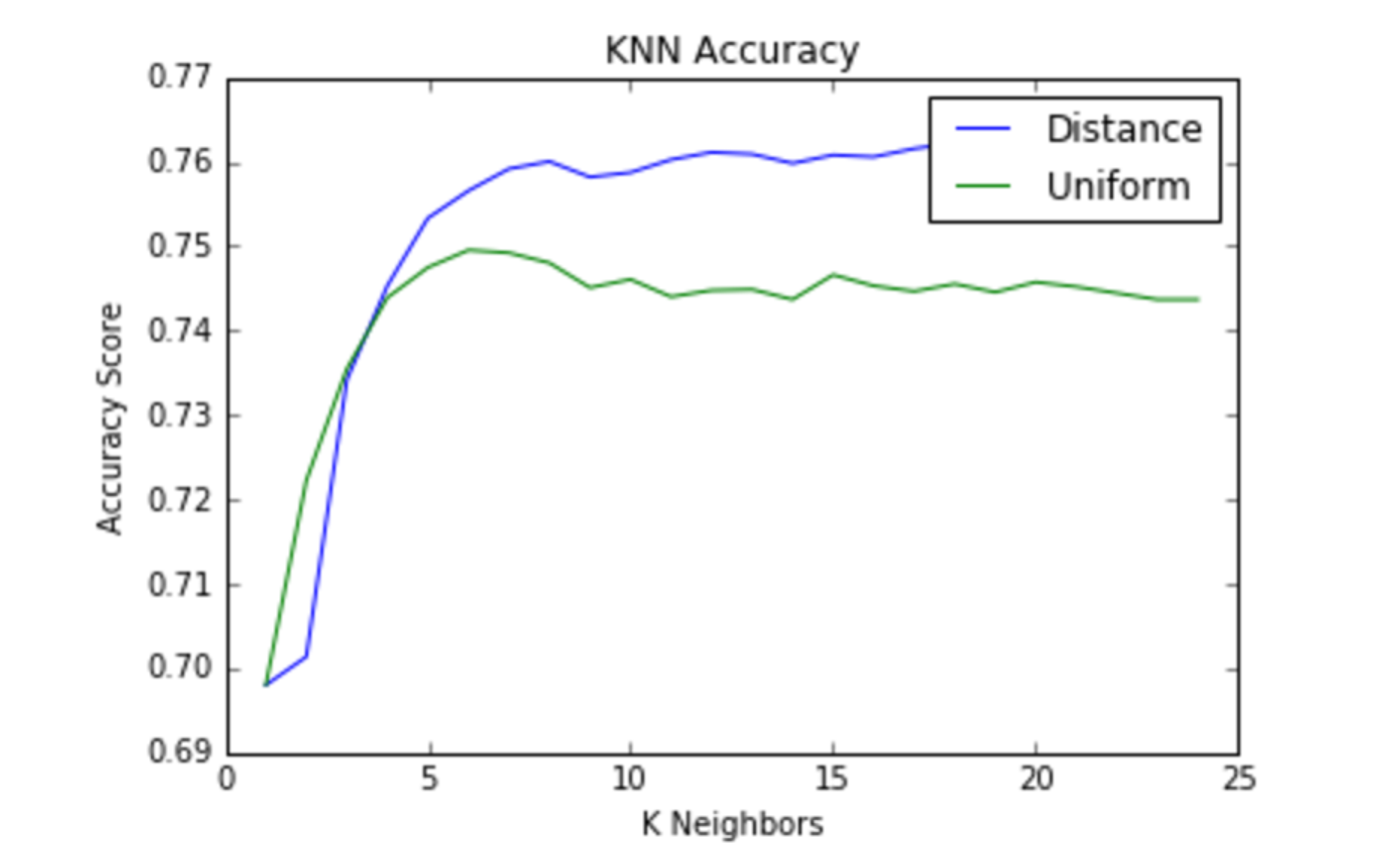
The approach did not work well at all. It got only .065% of the classifications correct. This is far below the number you would expect to get right if you randomly guessed. We believe this failure is due, in part, to the fact that the data is not evenly distributed across the various categories. A huge portion of the foods in the dataset was sold in France (74%), even though there were 59 unique countries in total. This presents an issue with our approach, because the classification predictions are not weighted to favor countries that appear more often. For example, if a test case most closely resembles a one country, but is also very close in distance to France, the most common choice, it might be a better guess to go with France.

To account for some of the skew, we re-tested this approach without France included. In other words, we eliminated all rows where the food was sold in France. This did not result in a dataset with balanced categories, but it was better, at least. For this test, the approach got 8% of the classifications correct. This is certainly not great, but at least it beats the number you’d expect to get right by random chance.

## K Nearest Neighbor Approach

We examined the same question, “can we classify which country a food was sold in based on its nutriment makeup” using the KNN approach. We thought this approach should fare better than the Nearest Centroid Classifier because it naturally has the weighting towards more common categories that was missing in the latter.

We explored which formula (distance or uniform) performed better, as well as the ideal number of neighbors to use. The graph below shows the accuracy results for both formulas across a range of neighbor values.



The best results were obtained using the distance measure and 20 neighbors. When we ran the KNN Classification with these parameters, we got 76% of the classifications correct. This beat the number that you would expect by chance significantly. Another method we used to evaluate how well the approach fared was how it compared to just guessing “France” every time, since France was by far the most common classification answer. Our KNN accuracy of 76% beat the “guess France” technique, but not by much. The “guess France” technique results in an accuracy of 74% for our particular train, test split. Interestingly, the KNN approach resulted in an accuracy of 96% on the training set. This suggests that perhaps the model was over-fit.

# kMeans clustering (kMeans\_Clustering.ipynb)

A feature that we found interesting in the dataset is "main\_category\_en". From what we could tell, that feature is a subjective classification for a particular food. We assume the person adding the food to the dataset would choose whatever they thought would fit as the main category for that food. This led us to ask whether or not we could group particular foods into categories based on nutriment values and we decided to do this in an unsupervised manner using kMeans clustering.

The first thing we did to prepare the data was to select 12 categories of food that had a sample size of over 1000. We did not want to try and classify into too many categories and we wanted enough data to make sure we didn't over-fit to outliers. Once we did that, we ended up with a group of 12 categories to use and we shrunk the full dataset to limit it to those 12 categories. Again we took all the nutriment values in that limited dataset as the data that we would train on. We converted the categories from strings to integers so we could compare the predicted clusters with the actual clusters. We trained the dataset using kMeans with an initialization method of k-means++ and iterated 20 times on the data. Once we fit the training set, we predicted the clustering of the training data and compared that with the clusters of the 12 unique categories.

The resulting completeness score was a measly 0.19 and the homogeneity score was also a low 0.17. This tells us that its difficult to cluster the foods based simply on nutriment values.