Master in Innovation and Research in Informatics (MIRI)

Machine learning course

Final Project

Open Food Facts

Cristina Capdevila Choy

Roberto Ruiz Wilson

Ana Giraldo Botero

Contents

Description of dataset and problem

The selected dataset gathers information related to the composition, characteristics and ingredients of food. The dataset was created in the aim of providing insights on nutrition facts related to food products from all around the world, the set used for this project was obtained from an already built site that seeks to deliver transparency to the consumer when purchasing processed food. In the Food Fact site (world.openfoodfacts.org/data), you can currently find applications to search particular facts on products from different countries and do advanced search based on some classification already done by the creators. We, on a different approach, will be focusing in the nutrition feature of 399,892 products of a selected subset, delivering a model to predict the rating/score of this particular product; so at the end the problem we will try to solve is to establish, accurately, whether a product is nutritional enough or not, according to the different features related to it.

Responsible variable:

- **nutriscore_score:** a score given to each product, based on how balanced its macronutrients are. Values from -15 up to 40.

Supplementary variables: 11 features

- **countries_en:** country of origin/commercialization of the product. 19 categorical values.
- **nutriscore_grade:** categorical/factor. An assigned letter from “a” to “e”.
- **additives_n:** number of additives found in nutrition value labels. Numerical value 1 to 24.
- **nova_group:** it is a kind of nutrition grade given to the foods, it is kept as a feature for extra info when grading the nutrition score. Numerical value from 1 to 4.
- **pnns_group_2:** categorical/factor correspond to the type of product, in an already defined list of groups of values
- **fat_100g:** fat supply of the product for a 100g portion.
- **carbohydrates_100g:** carbohydrate supply of the product for a 100g portion.
- **sugars_100g:** sugar supply of the product for a 100g portion.
- **fiber_100g:** fiber supply of the product for a 100g portion.
- **proteins_100g:** protein supply of the product for a 100g portion.
- **salt_100g:** salt addition per 100g of the product.

The feature selection is based on the main factors that, in general terms, define the nutritional value of any food: its chemical composition that is divided in the 3 main macronutrients: carbohydrates (fiber and sugar are types of it), proteins and fat. Some of the other features are included in order to analyze behavior of already created categories, classifications and groups.

Related previous work

The foodfacts site has different sections developed by an open community in multiple countries, there are more than one million products registered in the base with diverse information and that has been used to build applications of many kinds, they have also added to the site, some automated checks to guarantee the accuracy of the uploads as it is apparently done from all over the world. The site is divided in 2 main aparts: “Contribute” and “Discover” , the first section is where anyone can add products or complete information about the already existing ones, the most common contributors are consumers, producers, retailers, experts, but also here you can find a special section for people who want to do a more technic contribution on the differents projects and it is more related to management, design and development persea. In the second part you find an already built web application to do advanced search on products by name, brands, categories, origins and labels and with multiple criterias like specific brands, categories, labels, origins of ingredients, manufacturing places etc.

There is also a mobile Open Food Facts app that is completely open source and up on Github, the app was realised 7 years ago and makes it possible to scan food products to decrypt their labels, this year both the iPhone and Android versions were completely rewritten with the help of foundations and hackathon activities.

Data pre-process

Data cleaning

For the data cleaning process we performed a series of operations over the original dataset, which will be described below. These operations were done in Python due to the computational limitations of the R language. Once we were left with a dataset of a manageable size, the rest of the project was done in the R language. The data cleaning process consisted of:

1. Transform empty values (" ") or values marked as “unknown” to “NA”. Most of these empty values are actually zero values in the case of numeric variables. The value NA is used by R as the Null value.
2. Remove rows where responsive variable “nutriscore_score”, and categorical explanatory variable “country_en” is NA. We are unable to infer or calculate the values of the country from the already existing data, and nutriscore_score is used as the target variable so it would make no sense to have products without a score.

3. We removed all products from countries that have less than 1,000 products in the dataset. These countries are themselves outliers and in the more extreme cases those products were written in non-latin characters such as arabic that we cannot possibly interpret.
4. We kept only the rows of products with a single value for country ("country_en"), this was done to keep the country variable from becoming too complex. Products with more than one country represented only 1.6% of the dataset.

   Outcome: 399,891 rows.

Outlier Elimination

Visual Analysis

First visual exploration consists of identifying extreme outliers by graphically displaying data, detecting values out of a defined range (interval) and declaring as such.

-defined quantile: 0.975 (26.77667 distance).

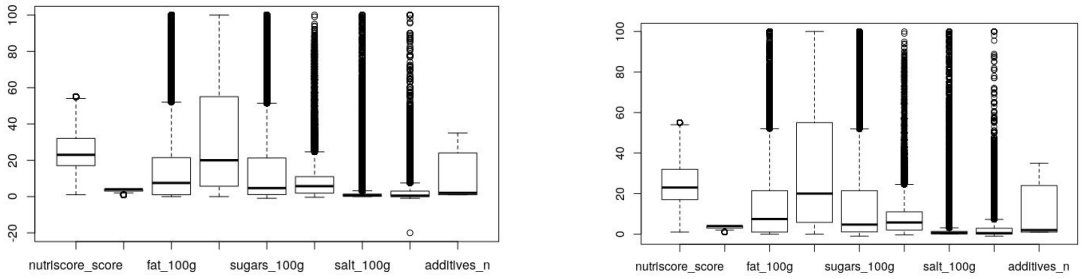

Figure 2: boxplot of variables vectors before and after outlier detection and input of missing values

Visualization and latent factors detection

In the first exploratory analysis we have considered all variables to be equally weighted to perform a standardized PCA and detect the most significant variables.

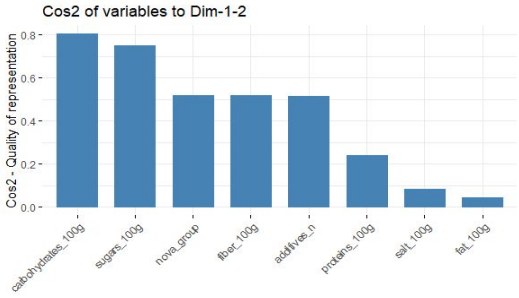

Figure 5: Percentage of total inertia explained by each variable.

The most well represented features are *carbohydrates_100g* along *sugars_100g*, interpreted from the plot; these two dimensions are closely related; explaining in a very similar way the same information about food. This can be due to the fact that both one and the other are usually taken as the same macronutrient in food, and in many cases the carbohydrate supply in one food is determined by the amount of sugar contained in it. Sugar is the simplest type of carbohydrate in its composition and it is frequently the most used one in food processing for being cheap and easy to get. Also inferred from the plot we could say that these two features are redundant, as one can be explained by the other and we will not lose significant information in the analysis when choosing only one of them for further analysis.

nova_group and *additives_n* are also very close in relation, as it is a general thought that additives are determinant in the nutrition grade given to a food, not in a good manner, thus additives are well known ingredients as toxic and harmful for your health. Therefore, either additives or nova group factor can also be removed from the data set.

One of the least well represented features is *salt_100g*, being highly represented only by the 4th component, it is also poorly correlated with carbohydrates, indicating that they measure food from opposite dimensions. This is very true when analyzing a food value as salt is by no mean a macronutrient but a mineral (also known as micronutrient), so it is less significant when grading a food, and its contribution to the nutrition score is low in proportion.

Both *protein_100g* and *fiber_100g* variables are close to dimension axes revealing some sort of importance in the significance of its values . Protein being explained mostly for component 1 and 3 and fiber for 2.

fat_100g on the other hand, being a macronutrient, should be more significant in the component and quality of representation, we do not hold any theory on why this should happen, but will have it as a flag in later analysis in the aim of finding further information related to help us find additional interpretation.

Clustering

When the hierarchical clustering is applied, the sum of squares starts out at zero (because every point is in its own cluster) and starts to grow as we merge the clusters. Ward's method is used to keep this growth as small as possible and it looks for clusters in multivariate Euclidean space.

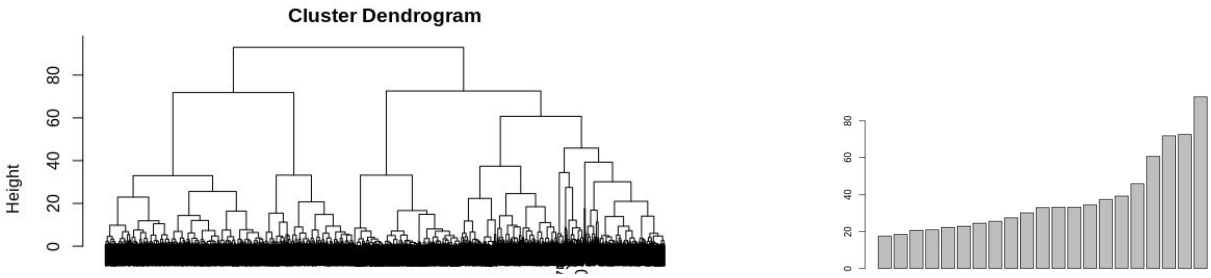

Figure 6: Dendrogram and boxplot of the 20 larger heights of the tree.

If we have a look at the dendrogram we see that there are two distinct groups: the left and the right one. To analyze this data set we have decided to take 5 clusters following the larger jump in heights thumb rule. We see that after the 4th division the height of the tree is having a smoothing drop and so we should take 5 clusters. Finally, we h ave defined the number of classes and assign the number of the elements that will contain each cluster by using the cutree method. We proceed to calculate the inertia between clusters and we get the 59.19%. The number of individuals per cluster are: 1140, 686, 769, 768 and 635.

The following 2-dimensions plot shows the projection of the data to the first factorial plane coloured by the cluster they belong to. We can see that all clusters overlap at the visual mass center of the data. The clusters seem to be quite well bounded. However, cluster 1 and 5 are crossing each other.

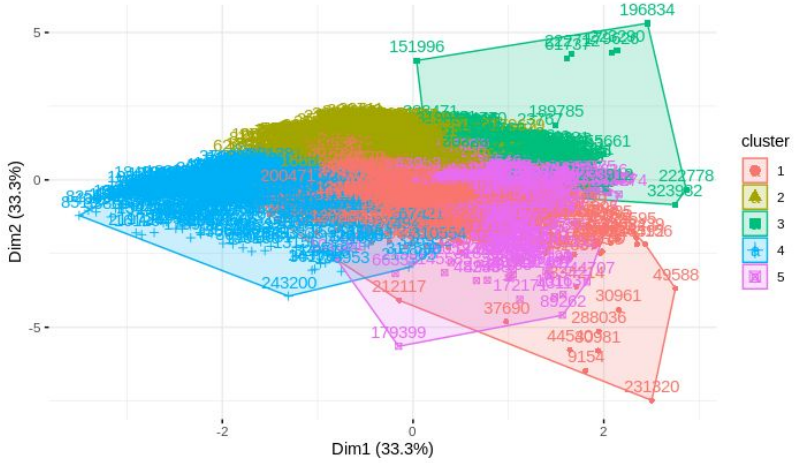

Figure 7: Clustering data projected on the first factorial plane.

To continue we perform the consolidation process, which means to run k-means algorithm. This process is utilized to improve the initial partition from hierarchical clustering. We expect to see some changes after consolidation. Below, we can see the old clustering to the left and the new clustering to the right (after consolidation).

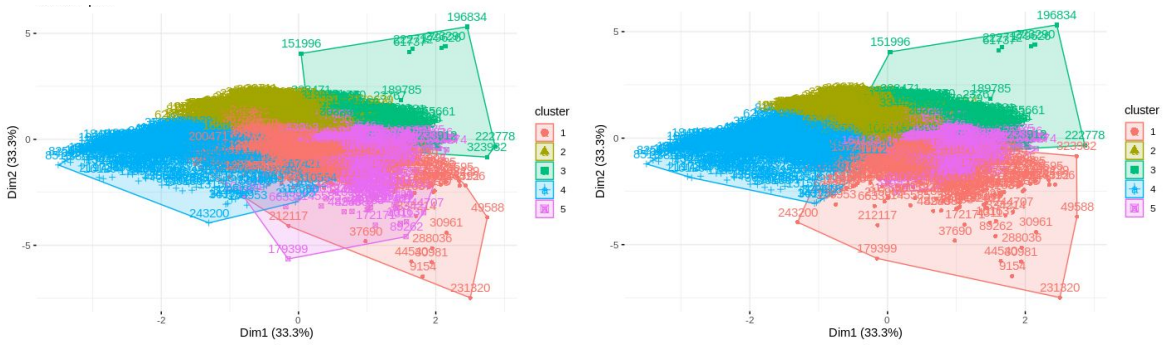

Figure 8: Clustered data before consolidation (left) and after consolidation (right).

As we see some clusters have completely changed shape because some individuals have joined them or have left from them to be part of the cluster they are more likely. Moreover, there has been an important change of the inertia between clusters, it has increased till 65.4%.

In addition, we get a higher value of the Calinski-Harabasz index after consolidation, compared to the index before consolidation. This means the clustering has been improved after the consolidation process. Before the consolidation we get 1424 and after we get 1856.

When performing a catdes we get a description of the different clusters by the variables. The results give us information about the p-value of the chi-square test used to evaluate the link between one qualitative variable and the qualitative variable corresponding to all the clusters. A p-value less than 5% means that the qualitative variable can explain the clusters. Also, we can see the results of the v.test, for example for the first cluster we have that *fiber_100g* with a high result is further in a positive direction from the overall mean. The opposite happens with *additives_n* that is significantly smaller than the overall mean.

The representation of the clustered data on the first factorial display is the following:

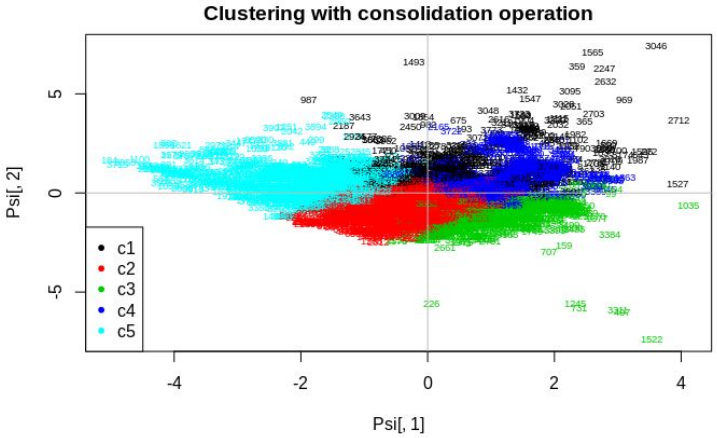

Figure 9: Clustering data projected on the first factorial display.

Based on the plot we can say that the 5th cluster is negatively correlated with the first dimension and smoothly positively correlated with the second dimension. The opposite behaviour is represented by the 3rd cluster, which is positively correlated with the first dimension and slightly negatively correlated with the second one. Then, the 4th cluster is also positively correlated with the first dimension and quite a bit with the second one. Similarly, the 1st cluster shows the same behaviour but not as strong as the 4th cluster. Finally, the 2nd cluster is having the opposite behaviour as the 1st one. It is showing a negative correlation with the first dimension and a little negative correlation with the second one. However, as in the 1st cluster, this behaviour is not so strong.

Regression Modeling

Linear multiple regression

First approach is to generate a linear regression with all explanatory variables in it and later on a mix of different combinations, based on partial obtained results. As we cannot guarantee response variable to follow a normal distribution we used the generalized linear model function, all built models listed below:

model	included vbles	mean square error
model1	4 - only macronutrients	mse: 179.7608
model2	all 10 explanatory	mse 45.98685
model3	8 - nutritional related	mse: 49.16144
model4	8 nutritional related and log (target value)	mse: 0.5182961
model5	model4 + polynomial on *nova_group*	mse: 0.5176696
model6	model4 + polynomial on *fiber_100g*	mse: 0.5168053

With the achieved results we will keep model4, as a simpler model is always preferable, having a final equation with 8 coefficients that all together explain the relationship between the nutritional facts and the grades for each food with the overall score given to that product.

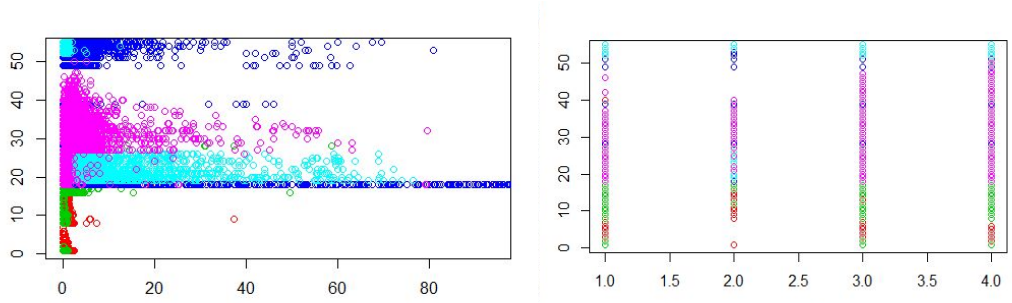

figure 10: *nutriscore_score* vs. *salt* and *nova_group* for each of the grades in *nutriscore_grade.*

Ridge Regression

Ridge regression is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This modification is done by adding a penalty parameter that is equivalent to the square of the magnitude of the coefficients, we will not be implementing this type of regression due to the characteristics of the majority of the coefficients in the already generalized linear regression.

Neural Network

We used an artificial neural network to predict the nutrition grade of the products of the DataSet (OpenFoodFacts). As before, we use the nutritional values as inputs of the models so as to predict the nutrition grade ($nutriscore_grade) First, we take a sample of the 1% of the original dataset for computational reasons, but still having a dataset of about 4000 instances. Then, we split the dataset into learning and test sets, selecting randomly 2/3 and 1/3 of the data. Finally, we standardize the input dataset and factorize the nutrition_grade attribute. We use the “nnet” library to compute the neural network.

To have a baseline reference and the first results, we fit a MLP without hidden neurons, i.e. a linear logistic model. We get training and test errors of 42.05% and 41.5%. We proceed by priorly building again the MLP but with 5 hidden layers. Therefore, we get a network of size 5x5x6 and 65 weights. Indeed, 65 weights mean: 6(input layers) x 5(hidden layers) + 5(input layers) x 5(hidden layers) + 5 bias (hidden layers) x 5 bias (output layers). The resulting weights are quite different in magnitude as we expected and the network might be so unstable. Hence we decide to use a decay value for the weights.

When we introduce a weight decay value of 0.5 and we get an error on the training and test sets of 29.57% and 31.43%. We see how this regularization parameter can help the MLP so we proceed to compute a cross-validation model of size 10x10 and solve the best combination of decay value and number of hidden layers.

We use the 10x10 cross-validation, CV, as a resampling method to find which accuracy we can get for the MLP by using different amounts of hidden layers. Thus, we declare the amount of hidden layers we want to test and perform the CV with each structure. We get the following (left table):

	size <dbl>	decay <dbl>	Accuracy <dbl>	Kappa <dbl>	AccuracySD <dbl>
1	2	0	0.6270	0.5229	0.03531
2	4	0	0.6801	0.5894	0.02862
3	6	0	0.7001	0.6156	0.02837
4	8	0	0.7107	0.6293	0.02713
5	10	0	0.7215	0.6432	0.02986
6	12	0	0.7241	0.6466	0.02746
7	14	0	0.7234	0.6456	0.02768
8	16	0	0.7222	0.6442	0.02986
9	18	0	0.7169	0.6375	0.02624
10	20	0	0.7168	0.6375	0.02754

	size <dbl>	decay <dbl>	Accuracy <dbl>	Kappa <dbl>	AccuracySD <dbl>
1	12	0.01000	0.7301	0.6541	0.02602
2	12	0.01585	0.7281	0.6516	0.02691
3	12	0.02512	0.7295	0.6535	0.02617
4	12	0.03981	0.7277	0.6511	0.02700
5	12	0.06310	0.7278	0.6513	0.02445
6	12	0.10000	0.7273	0.6505	0.02215
7	12	0.15849	0.7250	0.6472	0.02426
8	12	0.25119	0.7251	0.6472	0.02286
9	12	0.39811	0.7247	0.6463	0.02595
10	12	0.63096	0.7145	0.6326	0.02465

The best result we get is for 12 hidden layers, i.e. we do the same process again using this structure and varying the decay value on weights. As a result we get the previous right table.

We conclude with a network structure of 12 hidden layers and a decay value of 0.01 which give us an accuracy of 0.7301 on the training set. Finally, we proceed to predict the test set with this configuration and we get an error of 26.63% which is quite satisfying. Printing the confusion matrix we see that the predictions are coherent with the categories nature. The nutrition grades are ordered and so it makes sense that the confusion matrix is showing mostly confusion with grades closer in order to the right grade.

	Reference				
Prediction	a	b	c	d	e
a	205	45	11	2	5
b	16	76	22	2	3
c	13	37	178	47	6
d	1	2	23	302	53
e	4	2	12	43	201

Random Forest

We used a random sample of 10% of the dataset to make the process, especially the parameter estimation, computationally feasible. Of that, 10% will be used for validation and the remaining 90% is what used to train the model. For computational reasons we also removed the predictor pnns_groups_2 due to it being a categorical value with a high cardinality.

In order to find the best Random Forest model, we used a grid search on two of the hyperparameters: the number of trees and the number of variables to use. For the number of trees we tried values in increments of 10 starting from 10 and ending in 100, then from there in increments of 100 ending in 500. For the number of predictors we used all numbers from 1 to 8, taking into account that our dataset has 9 variables. A grid search learns a different random forest model for each combination of hyperparameter models.

We show an abridged sample of the results obtained after running the grid. We get ever increasing accuracy both by increasing the amount of variables and the amount of trees. The increase is more significant by using more variables than by creating more trees. It would make sense to attempt training with an even larger number of trees, but the increase in accuracy we see is very small and the processing time is already long at 500 trees.

Accuracy	Variables	nTrees
0.8103648	6	300
0.8175704	7	300
0.819605	8	300
0.8119472	6	400
0.8180225	7	400
0.8198875	8	400
0.8122015	6	500
0.8173161	7	500
0.8200288	8	500

Finally, we wanted to confirm the results seen actually generalize and are not the product of an overfit model. So we trained a random forest model with 8 variables and 500 trees and used it to predict the holdout data we reserved for testing. After doing so, we achieved an accuracy of 82.96% on the testing data. This is impressive given that it's even larger than the result we obtained on the training data and is strongly suggestive that our model did not overfit at all.

Given these results, it would be promising to try to fit a larger model with all of the dataset instead of only the 10% sample, and with more trees. We would need a more powerful computer than we have access to for this experiment.

Conclusions

- As a result of data redundancy first, identified in the principal component analysis, we were able to manipulate and maneuver different approaches when performing later actions over the data: Being an example of it the removal of a variable, for the tree modeling process, that we knew in advance was explained by another feature kept in the analysis.
- Generalized regression models provide a solid base when modeling data, as they deliver significant insights on the importance of the explanatory variables in the target one, their flexibility allows to have a good enough solution (even though no optima) for the prediction. In the process for linear modeling we were able to determine that the relation between the target variable and the responsible ones was not as not limited only to the nutritional facts of the ingredients, but also related to outer information of the product.
- The MLP shows us an acceptable accuracy when predicting the nutrition grade based on the nutritional values. We end up having an accuracy of 73.4% on the test dataset. The confusion matrix shows us that the results are coherent with the target values.
- The Random Forest model proved to be the most accurate than the Neural Network, with an 83% accuracy on the testing data. Thus proving that more classical Machine Learning methods are more appropriate for this type of prediction task. The Random Forest model also showed potential for even better results in further experiments with a more powerful computer.