Decision Models For The Nutri-Score Label Of Foods

Edoardo CONTE
Pratham SOLANKI

CentraleSupélec

January 19, 2020

Problem Statement

The Nutri-Score is a nutrition label that converts the nutritional value of products into a simple code consisting of 5 letters. We develop and test various decision models – additive, sorting and machine learning models that determine the Nutri-Score of a food given its various characteristics.

Figure 1: Nutri-score logo

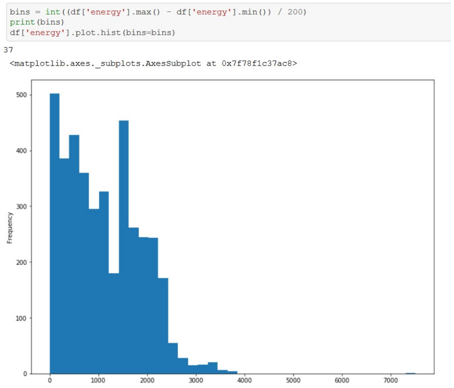
UTilites Additives method

- Implemented the UTA approach as taught in class
- Implemented custom algorithm to determine the intervals for each criteria

Train size	Test size
799	3196

Table 1: Dataset size for UTA approach

Determining the number of intervals for a criterion

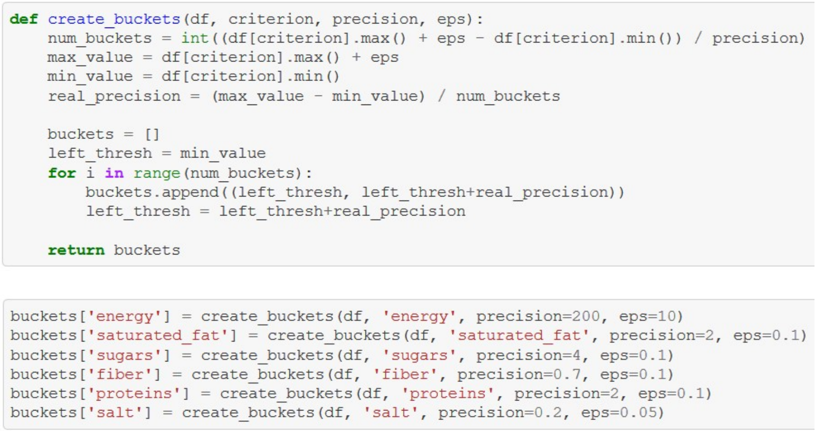


```
bins = int((df['energy'].max() - df['energy'].min()) / 200)
print(bins)
df['energy'].plot.hist(bins=bins)
```

```
37
<matplotlib.axes._subplots.AxesSubplot at 0x7f78f1c37ac8>
```

Figure 2: Bin histogram for Energy criterion with 200 precision

Creating intervals for a criterion



```
def create_buckets(df, criterion, precision, eps):
    num_buckets = int((df[criterion].max() + eps - df[criterion].min()) / precision)
    max_value = df[criterion].max() + eps
    min_value = df[criterion].min()
    real_precision = (max_value - min_value) / num_buckets

    buckets = []
    left_thresh = min_value
    for i in range(num_buckets):
        buckets.append((left_thresh, left_thresh+real_precision))
        left_thresh = left_thresh+real_precision

    return buckets
```

```
buckets['energy'] = create_buckets(df, 'energy', precision=200, eps=10)
buckets['saturated_fat'] = create_buckets(df, 'saturated_fat', precision=2, eps=0.1)
buckets['sugars'] = create_buckets(df, 'sugars', precision=4, eps=0.1)
buckets['fiber'] = create_buckets(df, 'fiber', precision=0.7, eps=0.1)
buckets['proteins'] = create_buckets(df, 'proteins', precision=2, eps=0.1)
buckets['salt'] = create_buckets(df, 'salt', precision=0.2, eps=0.05)
```

Figure 3: Creating intervals with optimal precision values

Result preference order on test set

Actual label	UTA score
A	0.00528304494
A	0.004561078976455346
D	0.0041243307900502
B	0.004084885049517742
A	0.0039165089854748605
D	0.0037278357439320753
D	0.0037084626995298863
C	0.0036896906677074295
D	0.002533260919903306
E	-0.004868497585240849

Table 2: Preference ordering for a sample of 10 foods from test set

Majority Rule sorting procedure

- Determining the limiting profiles using a learning model
- Manually setting the limiting profiles by studying the basic statistics behind the data

Train size	Test size
3196	799

Table 3: Dataset size for MR-sort

Learning the Limiting Profiles

	Energy	Saturated Fat	Sugars	Fiber	Proteins	Salt
π^6	0	0	0	100	100	0
π^5	594	1	2.1	4	8	1
π^4	669.1	2	3.1	3	7	2
π^3	1049.99	3.1	4.1	2	6	3
π^2	1873.99	11.1	24	1	5	4
π^1	7510	100	100	0	0	100

Table 4: Limiting Profiles learnt using Linear Programming

Learning the Limiting Profiles

To programmatically learn the limiting profiles we meticulously designed a system of equations. Precisely, for every food in our training data we constrained that the value for each criteria lies in-between the respective profile thresholds.
For example, if *food*1 has nutri-score label *A* then the value of its *energy* criterion should be between $\pi^6[energy]$ and $\pi^5[energy]$:

$$food1[energy] >= \pi^6[energy] \tag{1}$$

$$food1[energy] < \pi^5[energy] \tag{2}$$

Note that we are minimising *energy* i.e. the best food (label *A*) should have least *energy*.

Learning the Limiting Profiles

Next, in-order to write an objective function we incorporate the concept of errors. A resultant subset of equations for *food*1 with label *A* and for *energy* criterion is as follows:

$$Minimize\ \epsilon^{6}_{food1}[energy] + \epsilon^{5}_{food1}[energy] \tag{3}$$

$$food1[energy] >= \pi^{6}[energy] - \epsilon^{6}_{food1}[energy] \tag{4}$$

$$food1[energy] < \pi^{5}[energy] + \epsilon^{5}_{food1}[energy] \tag{5}$$

$$\pi^{6}[energy] = 0 \tag{6}$$

$$\pi^{5}[energy] = maxEnergy \tag{7}$$

$$\pi^{6}[energy] < \pi^{5}[energy] \tag{8}$$

Learning the Limiting Profiles

For a more comprehensive understanding, following are the set of equations that concern the same food but for a different criterion (*proteins*) which needs to be maximized:

$$Minimize\ \epsilon^6_{food1}[proteins] + \epsilon^5_{food1}[proteins] \tag{9}$$

$$food1[proteins] <= \pi^6[proteins] + \epsilon^6_{food1}[proteins] \tag{10}$$

$$food1[proteins] > \pi^5[proteins] - \epsilon^5_{food1}[proteins] \tag{11}$$

$$\pi^6[proteins] = maxProteins \tag{12}$$

$$\pi^5[proteins] = 0 \tag{13}$$

$$\pi^6[proteins] > \pi^5[proteins] \tag{14}$$

Similarly we do this for all the foods in the training set, for all criteria to obtain a system of equations and an objective function which we solve using linear programming.

Manually setting Limiting Profiles

	Energy	Saturated Fat	Sugars	Fiber	Proteins	Salt
π^6	0	0	0	100	100	0
π^5	1205.1464	1.1531	8.1063	5.4	20	0.3882
π^4	1446	2.4919	13.3546	5.3	19	0.5467
π^3	1663.6701	7.0429	22.0533	5.252	18.3499	1.2891
π^2	2009.9287	13.6655	33.3945	4.6768	13.762	2
π^1	7510	100	100	0	0	100

Table 5: Manually set Limiting Profiles

Accuracy Comparsion

Learnt Profiles	Manual Profiles
0.4355	0.3854

Table 6: Test Accuracy comparison of MR-sort rule

Machine Learning

We have implemented three different Machine Learning algorithms to predict the Nutri-scores – Decision Tree, Random Forest and Gradient Boosting (XGBoost).

Train size	Validation size	Test size
3595	200	200

Table 7: Dataset size for Machine Learning algorithms

Decision Tree	Random Forest	XGBoost
0.825	0.865	0.855

Table 8: Test Accuracy comparison of various ML models

Conclusion

- With our configuration, the Nutri score cannot be explained by an additive or sorting model
- Decision Tree and corresponding ensemble models are able to approximate the Nutri score better