Decision Models For The Nutri-Score Label Of Foods

Edoardo CONTE
CentraleSupélec
edoardo.conte@live.it

Pratham SOLANKI
CentraleSupélec
pratham.c.solanki@gmail.com

January 19, 2020

Abstract

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.

Keywords decision-modeling · additive-models · uta · majority-rule-sorting · machine-learning · decision-tree · random-forest · gradient-boosting · xgboost

1 Introduction

Figure 1: Nutri-score logo

In France, the Nutri-Score logo (see Figure 1) was elaborated by Santé publique France, a department of the Health Ministry, based on the scientific works of Professor Serge Hercberg (University Paris 13) and the experts of ANSES (Agence nationale de sécurité sanitaire de l'alimentation, de l'environnement et du travail), another department of this ministry. The details behind the calculation of Nutri-Score is available on the official website[1].
The aim of this project is to develop and test three different kinds of decision models to determine the Nutri-Score of a food. We develop UTilités Additives model3, Majority rule sorting model4 and Machine learning model5. Our goal is to conclude whether or not these models are able to explain the real Nutri-Scores.
This project is hosted on GitHub[2].

2 Data preprocessing

Before implementing any models we first did some data pre-processing and cleaning:

- Removed all the extra columns other than the required features and the label – $energy, saturated\_fat, salt, sugars, proteins, fiber$ and $nutri\_score\_label$.

[1] https://nutriscore.colruytgroup.com/colruytgroup/en/about-nutri-score/
[2] https://github.com/PrathamSolanki/nutri-score-decision-models/

- Removed all the rows with missing values. After doing so the data has a total of 3995 foods.
- Encoded the nutri score labels to integer values.

3 UTA Approach

In this section we aim to rank various foods using the UTA approach. The dataset size for this section is shown in Table 1.

Train size	Test size
799	3196

Table 1: Dataset size for UTA approach

We used a custom logical method to determine the number of intervals and to create the intervals for each criteria. We visualize the bin histograms for every criteria with many precision values to find the best precision and therefore the optimal number of intervals for each criterion. As shown in the Figure 2, we find the best precision for $energy$ criterion to be 200 and the optimal number of intervals to be 37. We highly recommend to view the Python notebook on our Github repository to understand how we determine the intervals for each criteria.

Next, we create the determined number of intervals for each criteria using the respective optimal precision values as shown in Figures 3 and 4.

```
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
```

Figure 3: Function to create intervals given a criterion and precision

```
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 4: Creating intervals with optimal precision values

After we have created the intervals for each criteria, we form a system of equations using the training data (799 foods) for an UTA approach exactly as taught in class (as shown in lecture slides for chapter 4). Then, we use linear programming to solve this system of equations and learn the marginal utility

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

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

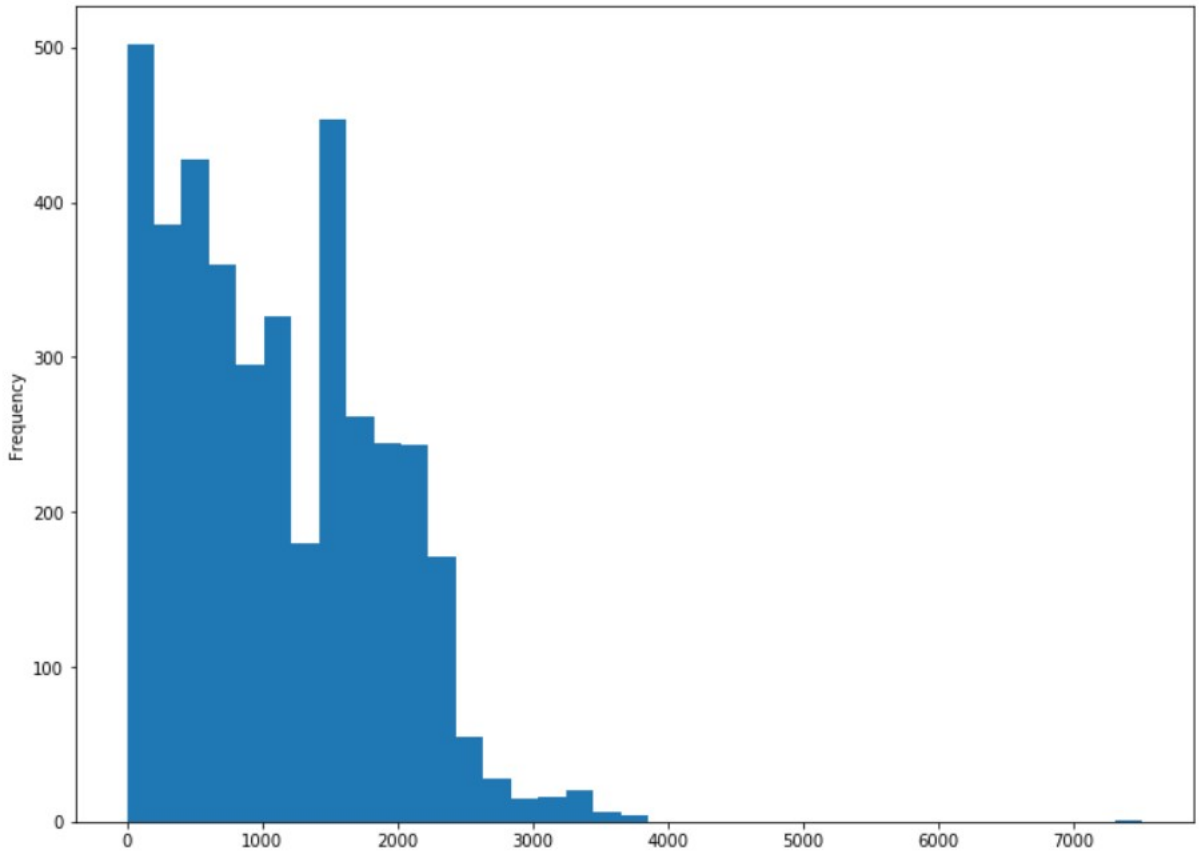

Figure 2: Bin histogram for Energy criterion with 200 precision

functions associated with each criterion. The marginal utility functions for each criteria are shown in Figures 5, 6, 7, 8, 9 and 10. Thereafter we assign scores to the foods in test data using the learnt utilities and sort them accordingly. A resultant preference of 100 foods from the test data is presented in a separate file **AdditiveNutriScoreResults.csv**.

3.1 Analysis

We can see from the preference result that the preference order is not perfect. For a better understanding we present a smaller preference order of 10 foods from test data in Table 2. The order is inaccurate. We thus conclude that our implementation of UTA approach is unable to perfectly approximate the real Nutri score.

4 Majority Rule sorting procedure

In this section we aim to classify the foods using the MR-Sort[5] procedure. In-order to do so we first need to set the limiting profiles. We have experimented both by determining the profiles using a learning

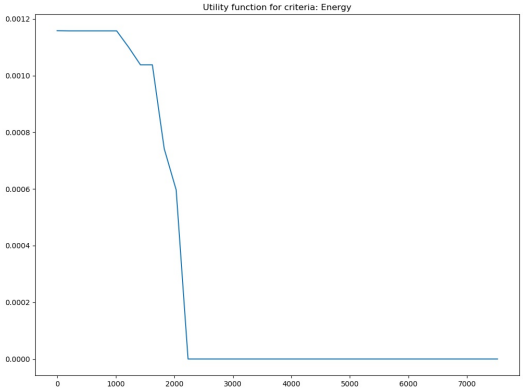

Figure 5: Marginal utility function for energy criterion

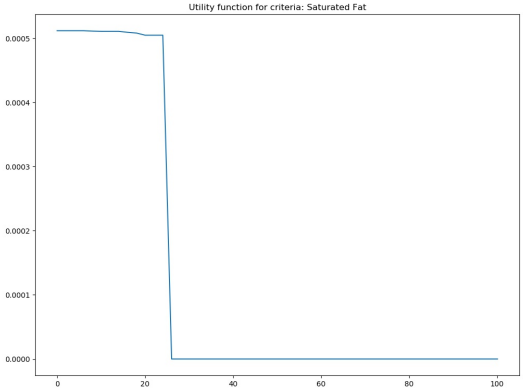

Figure 6: Marginal utility function for saturated fat criterion

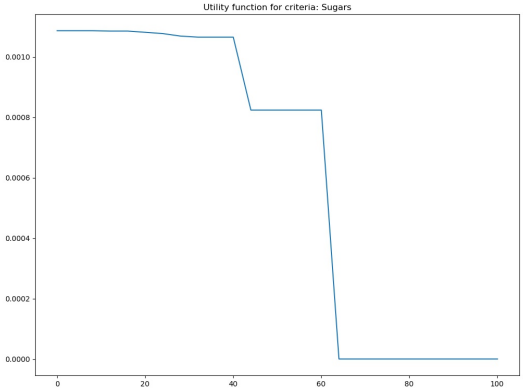

Figure 7: Marginal utility function for sugars criterion

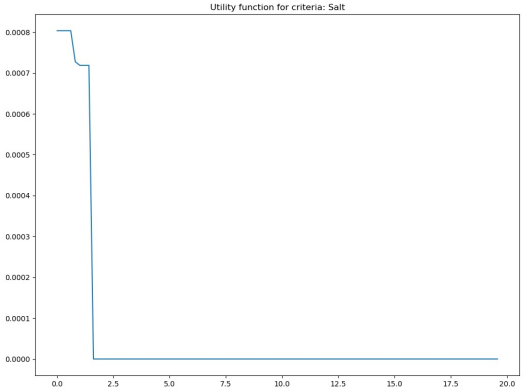

Figure 8: Marginal utility function for salt criterion

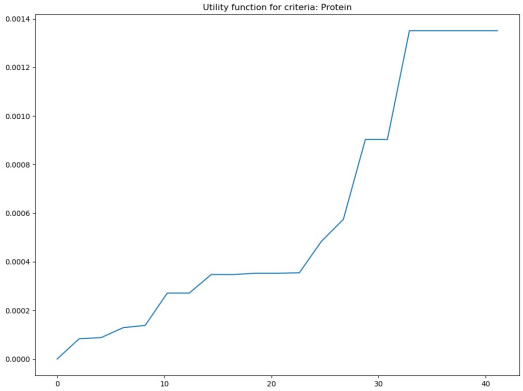

Figure 9: Marginal utility function for proteins criterion

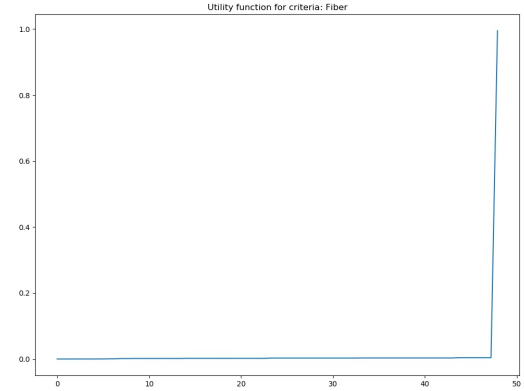

Figure 10: Marginal utility function for fiber criterion

Actual label	UTA score
A	0.00528304494
A	0.004561078976455346
D	0.00412433307900502
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

model as well as manually setting the profiles by studying the basic statistics behind the data (shown in Table 6).
The dataset size for this section is shown in Table 3.
We have implemented our own version of MR-Sort procedure where we categorize the foods a bit differently. Initially, when we were following the traditional approach of categorizing foods (using the pessimistic and optimistic approaches) one of the main difficulties we faced was that many foods were unable to clear the threshold for any level and thus they were not categorized into any label. To overcome this we came up with a slightly modified approach where we categorize a food into that label for which its weighted sum is maximum. This way we categorize each and every food regardless of whether it clears a threshold or not. Also with this approach the classification result does not depend upon what version of MR-Sort (Pessimistic or Optimistic) we are implementing. Hence, we have a single function for classification.

Train size	Test size
3196	799

Table 3: Dataset size for MR-sort

4.1 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

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 $food1$ has nutri-score label A then the value of its $energy$ criterion should be between $\pi^6[energy]$ and $\pi^5[energy]$:

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

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

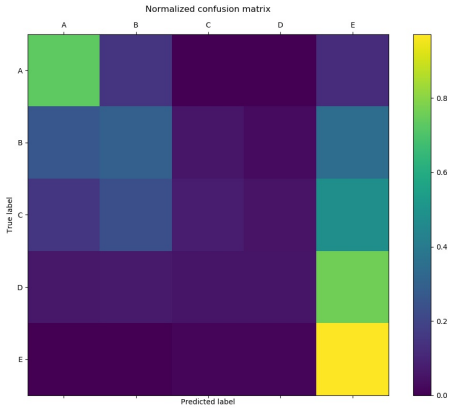

Figure 11: Normalized Confusion Matrix for MR-sort rule, with learnt limiting profiles

Note that we are minimising $energy$ i.e. the best food (label A) should have least $energy$. Next, in-order to write an objective function we incorporate the concept of errors. A resultant subset of equations for $food1$ with label A and for $energy$ criterion is as follows:

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

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

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

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

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

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

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. The learnt profiles are shown in Table 4.

Once we have the limiting profiles we use the MR-Sort procedure to classify foods from the test set and calculate the classification accuracy. The MR-Sort procedure with learnt limiting profiles yields an accuracy of 0.4355. The normalized confusion matrix is shown in Figure 11.

	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

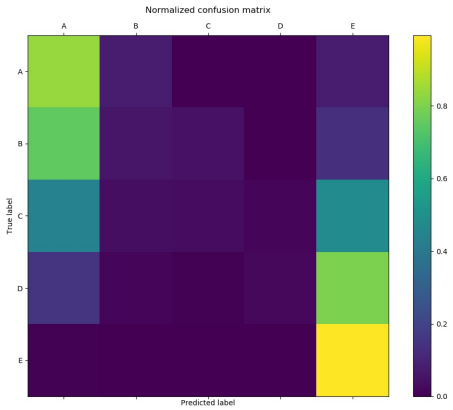

Figure 12: Normalized Confusion Matrix for MR-sort rule, with manually set profiles

4.2 Manually setting Limiting Profiles

To set the profiles manually we first studied the basic statistics of our data (shown in Table 6). As we can see there is no definite range of values for any criteria, for any score making it very difficult to set an appropriate profile. We experimented with a couple of thresholds at each level – max, min but the we get highest accuracy when the profile thresholds are set equal to the sum of mean and standard deviation. The manually set profiles are shown in Table 5.
Using these profiles the MR-Sort procedure yields an accuracy of 0.3854. The normalized confusion matrix is shown in Figure 12.

4.3 Analysis

A summary of accuracy comparisons of both the experiments with MR-Sort procedure are shown in Table 7. The learnt limiting profiles are certainly better than the manually set profiles. However, both accuracies are unacceptable, they are even less than 50% which basically concludes that the real Nutri-Scores cannot be explained by our implementation of a simple sorting model. To back our claim we show the data statistics (Table 6) where there is no definite range in the values of criteria for a particular score. And as the main idea behind the MR-Sort procedure is to classify using a limiting profile (which specifies the range in values of each criteria) it is very difficult for our simple sorting model to yield a high accuracy.

Energy				
Score	Maximum	Minimum	Mean	Standard Deviation
A	2575.0	0.0	643.7117270788913	561.4347337942038
B	2920.0	0.0	678.0013333333333	514.6467395302051
C	3464.0	0.0	984.2051282051282	679.4650271736979
D	3698.0	0.0	1414.8306451612902	595.0981548225855
E	7510.0	117.0	1918.4272445820434	682.5797509477326
Saturated Fat				
Score	Maximum	Minimum	Mean	Standard Deviation
A	100.0	0.0	15.454705882352949	11.523354465190025
B	42.0	0.0	7.210897177419362	6.454610278407766
C	60.0	0.0	2.8419413919413934	4.201056278455641
D	8.9	0.0	1.2675683333333334	1.2244226694458804
E	6.2	0.0	0.4827775053304908	0.6704036313106108
Sugars				
Score	Maximum	Minimum	Mean	Standard Deviation
A	84.0	0.0	28.28348297213622	19.799645939333487
B	100.0	0.0	15.136774294354842	18.257789857051566
C	71.0	0.0	9.192344322344333	12.860959094292403
D	67.0	0.0	5.066904999999999	8.28770104937318
E	37.0	0.0	3.6324946727078844	4.473808247513058
Fiber				
Score	Maximum	Minimum	Mean	Standard Deviation
A	23.0	0.0	2.2782352941176467	2.3985778694288107
B	41.0	0.0	2.2508972782258043	3.0011460393133604
C	27.0	0.0	2.4200708180708164	2.8043261462146223
D	12.8	0.0	2.123516666666667	2.2188552598496574
E	47.8	0.0	3.7477484008528803	4.160338401162896
Proteins				
Score	Maximum	Minimum	Mean	Standard Deviation
A	33.3	0.0	7.446687306501548	6.315374977154655
B	41.0	0.0	10.000453629032265	8.349511444879901
C	29.0	0.0	7.215140415140413	6.57354063669378
D	27.0	0.0	6.515533333333333	5.512884472539595
E	32.0	0.0	6.353315565031989	5.851187182951755
Salt				
Score	Maximum	Minimum	Mean	Standard Deviation
A	8.0	0.0	0.3898036928792569	0.6396938718702941
B	8.0	0.0	0.49610492491028246	0.5871957450532012
C	19.52	0.0	0.4413781167277166	0.8477457611061571
D	2.64	0.0	0.29509828533333354	0.2516661428052885
E	5.2	0.0	0.15859644190831593	0.22964358363603268

Table 6: Data Statistics

Learnt Profiles	Manual Profiles
0.4355	0.3854

Table 7: Test Accuracy comparison of MR-sort rule

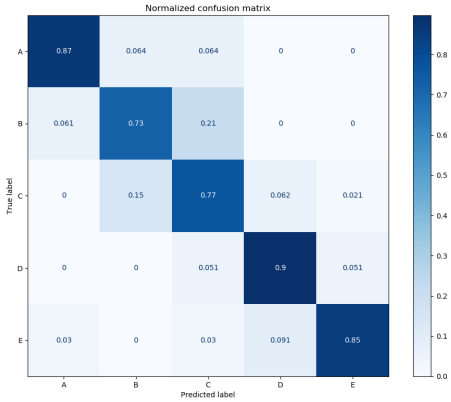

Figure 13: Normalized Confusion Matrix for Decision Tree Classifier

5 Machine Learning

We implement three different Machine Learning algorithms to predict the Nutri-scores – Decision Tree[4], Random Forest[1] and Gradient Boosting[3] (XGBoost[2]).
The dataset size for this section is shown in Table 8.

Train size	Validation size	Test size
3595	200	200

Table 8: Dataset size for Machine Learning algorithms

5.1 Decision Tree

We first implemented a Decision Tree classifier. It is one of the most basic machine learning classifiers and hence a good starting point. Also, Decision Trees are interpretable which is always a positive.
The resultant Decision Tree after running the algorithm is presented in a separate file **NutriScoreDecisionTree.pdf**. As it can be seen in the file, the decision tree is very big and has a large number of branches. This is because all the feature characteristics of foods are continuous (and not categorical) hence facilitating a large variety of splits. Therefore even though the decision tree is interpretable it is very tedious to comprehend the entire tree.
The decision tree classifier yielded an accuracy of 0.825 on the test set. The normalized confusion matrix is shown in Figure 13.

5.2 Random Forest

Having implemented a decision tree we explored a couple of ensemble learning methods. Random Forest is Bootstrap aggregating type of ensemble method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees.
The random forest classifier yielded an accuracy of 0.865 on the test set. The normalized confusion matrix is shown in Figure 14.

5.3 Gradient Boosting

We make use of the XGBoost library which implements the Gradient Boosting framework.
The XGBoost model yielded an accuracy of 0.855 on the test set. The normalized confusion matrix is shown in 15.

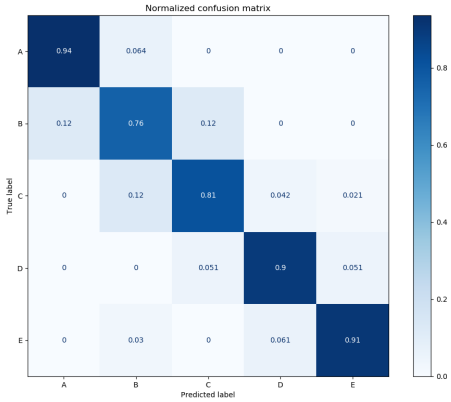

Figure 14: Normalized Confusion Matrix for Random Forest Classifier

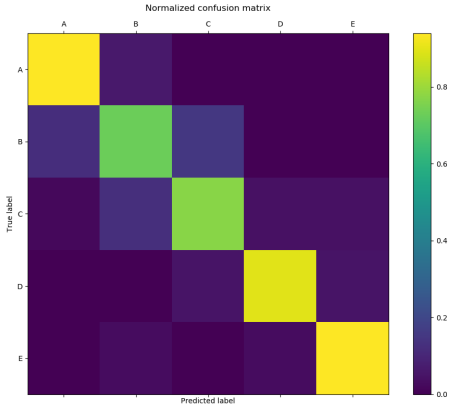

Figure 15: Normalized Confusion Matrix for XGBoost Classifier

5.4 Analysis

A summary of accuracy comparisons of the three machine learning classifiers are shown in Table 9. As we can see, the ensemble methods perform better than a single decision tree. Random Forest performs just slightly better than XGBoost.
Also, the machine learning models perform significantly better than the additive and sorting models.

Decision Tree	Random Forest	XGBoost
0.825	0.865	0.855

Table 9: Test Accuracy comparison of various ML models

6 Conclusion

Based on the results we obtained after carrying out a variety of experiments we conclude the following:

- With our configuration and implementation, 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.

References

[1] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. issn: 1573-0565. doi: 10.1023/A:1010933404324. url: https://doi.org/10.1023/A:1010933404324.

[2] Tianqi Chen and Carlos Guestrin. “XGBoost”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16* (2016). doi: 10.1145/2939672.2939785. url: http://dx.doi.org/10.1145/2939672.2939785.

[3] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *Annals of Statistics* 29 (2000), pp. 1189–1232.

[4] J. R. Quinlan. “Induction of decision trees”. In: *Machine Learning* 1.1 (Mar. 1986), pp. 81–106. issn: 1573-0565. doi: 10.1007/BF00116251. url: https://doi.org/10.1007/BF00116251.

[5] Olivier Sobrie, Vincent Mousseau, and Marc Pirlot. “Learning a Majority Rule Model from Large Sets of Assignment Examples”. In: *Algorithmic Decision Theory*. Ed. by Patrice Perny, Marc Pirlot, and Alexis Tsoukiàs. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 336–350. isbn: 978-3-642-41575-3.