Machine Learning on the Obesity Data Set

Tianqi Tang[1]

[1]Brown University
[1]Github Link: https://github.com/TianqiT/data1030-fa20-project

December 2020

1 Introduction

The topic of this project is obesity. This is an important topic because obesity affects many aspects of our lives. The data set[1] in use has 16 features and 1 target variable, NObeysdad. The target variable represents categorical obesity levels consisting of Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III. The problem that I will work on is categorical and is about making predictions about a person's obesity level based on other features. The 16 features are Gender, Age, Height, Weight, family history with overweight, Frequent consumption of high caloric food (FAVC), Frequency of consumption of vegetables (FCVC), Number of main meals (NCP), Consumption of food between meals (CAEC), SMOKE, Consumption of water daily (CH2O), Calories consumption monitoring (SCC), Physical activity frequency (FAF), Time using technology devices (TUE), Consumption of alcohol (CALC), Transportation used (MTRANS) and NObeyesdad. There are 2111 data points with no missing data. A brief summary of data values is presented below:

1. Gender: Male: 1068, Female: 1063
2. Age(years): min: 14, max: 61, mean: 24.31
3. Height(m): min: 1.45, max: 1.98, mean: 1.70
4. Weight(kg): min:39, max: 173, mean: 86.59
5. Family history with overweight: Yes: 1726, No: 385
6. FAVC: Yes: 1866, No: 245
7. FCVC: min: 1.0, max: 3.0, mean: 2.42
8. NCP: min: 1.0, max: 4.0, mean: 2.69

9. CAEC: No: 51, Sometimes: 1765, Frequently: 242, Always: 53
10. SMOKE: Yes: 44, No: 2067
11. CH2O: min: 1.0, max: 3.0, mean: 2.01
12. SCC: Yes: 96, No: 2015
13. FAF: min: 0.0, max: 3.0, mean: 1.01
14. TUE: min: 0.0, max: 2.0, mean: 0.66
15. CALC: No: 639, Sometimes: 1401, Frequently: 70, Always: 1
16. MTRANS: Public Transportation: 1580, Automobile: 457, Walking: 56, Motorbike: 11, Bike: 7
17. NObeyesdad: Insufficient Weight: 272, Normal weight: 287, Overweight Level I: 290, Overweight Level II: 290, Obesity Type I: 351, Obesity Type II: 297, Obesity Type III: 324

Many continuous features do not have units because they initially assumed categorical values and then preprocessed during the data set generation. The previous works using this data set includes Obesity Level Estimation Software[2] exploring 3 different methods for data set modeling. It was found that Decision Trees (J48) achieved the best results based on metrics Precision, recall, TP and FP rate. The authors also built a software using J48 and achieved 97.4% precision.

2 Exploratory Data Analysis

In this data set, every feature is identified as either categorical or continuous based on whether it contains 10 or less distinct values. Several figure are then created for data visualization. Here are some interesting examples.

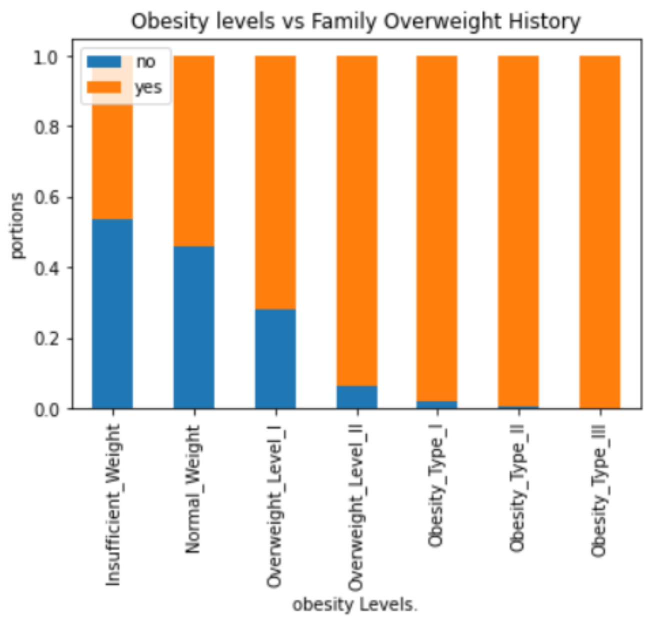

Figure 1: This figure shows how data points corresponding to different obesity levels are distributed. For each level, the bar is divided into two parts with the orange parts indicating people having family overweight history whereas the blue parts show the opposite case.

An interesting observation about Figure 1 is that people with insufficient weights or normal weights have higher probability of having no family overweight history. Moreover, people with moderate, rather than high obesity levels have the greatest chance of having overweight family members.
Another comparison is between age and daily length of technological devices usage, Figure 2.

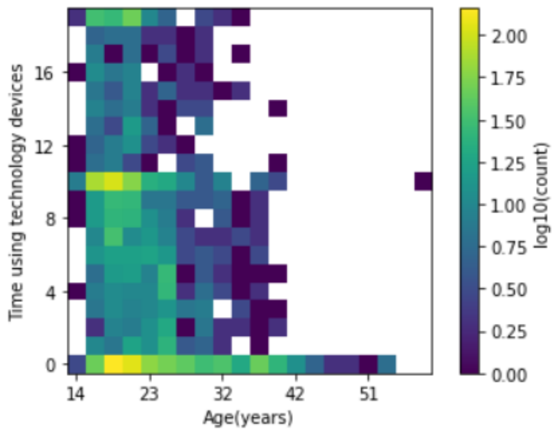

Figure 2: This figure shows how frequently people have technological devices usage based on their age. Many data points assume integer y-values.

We notice from Figure 2 that people tend to use less technological devices such as cell phones, TVs, etc., as they grow older.
One more comparison sets between alcohol consumption frequency and water intake in Figure 3.

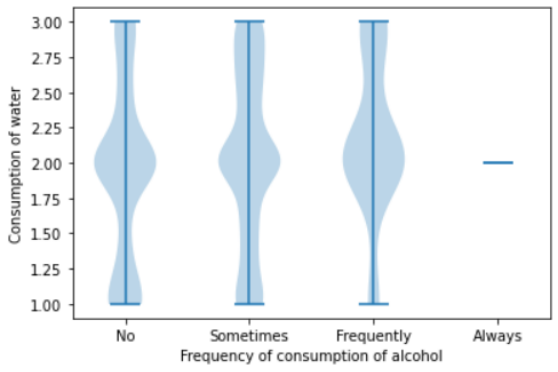

Figure 3: This figure shows how much water people drink based on their alcohol consumption. There's only 1 data point for the person who always drinks alcohol.

It can be seen that there's a severe data imbalance as people who classify as always drinking alcohol are few. Furthermore, for people who never or sometimes drink alcohol, some of them also drink little water whereas it is rare for people who frequently drink alcohol to drink little amount of water.

3 Methods

3.1 Preprocessing

Because this data set is not very large, I will ration 20% of data for testing. For the rest of 80%, I will ration another 20% of them for validating. Therefore, I use $64\%, 16\%$, and 20% of data for training, validating, and testing, respectively. There are several imbalanced features, but MTRANS is quite severe so I stratified splitting based on this feature. There's only one person who always drink alcohol, so no matter how the data is split, the imbalance of that will occur, so I did not stratified split based on alcohol consumption. I suggest splitting the data in a similar fashion based on these concerns. Since each data point contains information about a unique person who are independent of others, my data is IID with no group structure. It also does not involve a time series. I used 3 preprocessors: OrdinalEncoder, One-HotEncoder and StandardScaler and LabelEncoder on my Data. For categorical non-target features, only CAEC and CALC have ordered structures, so they will be processed by OrdinalEncoder, while the rest of them will be processed by One-HotEncoder. Since StandardScaler is always suitable for continuous features, all of the continuous features will be preprocessed by it. The target variable is categorical and ordered. Therefore, it will be preprocessed by LabelEncoder. Here's a summary of preprocessor usage on features.

1. OrdinalEncoder: CAEC, CALC
2. One-HotEncoder: Gender, Family History with Overweight, SMOKE, SCC, MTRANS
3. StandardScaler: Age, Height, Weight, FAVC, FCVC, NCP, CH2O, FAF, TUE
4. LabelEncoder: NObeyesdad

After preprocessing, there are 25 features and 1 target variable.

3.2 Evaluation Metrics

First, let count the number of instances of the 7 classes of the target variable. We can see from Table 1 that the target variable is not imbalanced.

I use 6 metrics: Accuracy, precision, recall, and 3 f_β scores for $\beta = 0.5, 1, 2$. Because the target variable is multi-class, each score, except accuracy, is calculated per class, then weighted by the abundance of entries of each class in the training set.

classes	counts
Insufficient_Weight	272
Normal_Weight	287
Overweight_Level_I	290
Overweight_Level_II	290
Obesity_Type_I	351
Obesity_Type_II	297
Obesity_Type_III	324

Table 1: 7 classes in the Target variable and their respective counts.

3.3 Machine Learning Techniques

For this classification problem, I use 3 algorithms: Logistic Regression (LR) with Elastic Net penalty, Random Forest (RF) and XGBoost. For the sake of baseline calculation, I also train a dummy classifier using the "most_frequent" setting, *i.e.*, it picks the most frequent class in the training set to be the predictions for all test data. I do not choose Support Vector Machine (SVM) as a potential algorithm because this data set is not too small and SVM may take a long time training the data.

3.4 Hyperparameter Tuning

I tune the parameters of the algorithms using grid search. Here are the 3 lists documenting the parameters to be tuned and their values.
Parameters for Elastic Net LR:

1. C: [0.01, 0.025, 0.063, 0.16, 0.40, 1, 2.51, 6.31, 15.85, 39.81, 100.00]
2. l1_ratio: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

Parameters for RF:

1. max_depth: [10, 30, 100, 300]
2. max_features: ['auto', 'sqrt', 'log2']
3. min_samples_split: [16, 32, 64, 128]
4. min_samples_leaf: [1, 2, 4]

Parameters for XGBoost:

1. alpha: [0.01, 0.1, 1, 10, 100]
2. lambda: [0.01, 0.1, 1, 10, 100]
3. max_depth: [30,100]

I use Elastic Net penalty for LR because it is very versatile as it can be considered as a combination of Ridge and Lasso. For RF, it is important to tune max_depth because the model may underfit if max_depth is too low or overfit if max_depth is too high. For XGBoost, tuning alpha and lambda also finds the sweet-spot between underfitting and overfitting.
Regarding model randomness, because RF and XGBoost is non-deterministic, when I specify the ML algorithm to be RF or XGBoost, I set its random state to be the same as what I use in data splitting. I run all algorithms multiple times with predefined random states to ensure that each algorithm produces the same results across multiple runs when given the same random state.
To visualize the uncertainties emerged from random processes such as data splitting and non-deterministic fitting, I run each algorithm 10 times with distinct random states and collect their scores each time. The accuracy scores over the 10 runs are shown in Figure 4.

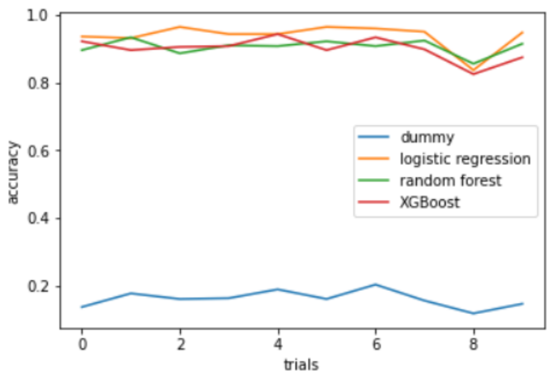

Figure 4: Accuracy scores for all algorithms over 10 independent runs.

To choose the optimal parameters for each model, I looked into the results in the 10 runs. Each run produces a best result for that run. Then for each algorithm, I select the most common best combination to be the estimator parameters.
To determine which model I should ultimate choose, I compute the averages for all of the scores over the 10 runs. The average scores are presented in Table 2.

It can be seen that LR consistently outperforms other algorithms in terms of average scores. Therefore, I choose LR to be my ultimate choice of machine learning algorithm for this data set.
The main metric that I will primarily consider to judge the performance of an algorithm is accuracy. I think it is a very standard metric and is widely applicable to most of the classification problems.

	Dummy	LR	RF	XGBoost
Accuracy	0.161	0.938	0.896	0.900
Precision	0.027	0.939	0.899	0.914
Recall	0.161	0.938	0.896	0.900
$f_{0.5}$	0.032	0.938	0.897	0.901
f_1	0.045	0.938	0.895	0.894
f_2	0.080	0.938	0.895	0.895

Table 2: Different average scores for the Dummy (baseline) and 3 ML algorithms.

4 Results

The optimal set of parameters for each algorithm is listed below.
Best parameters for LR:

1. C: 6.31
2. l1_ratio: 1

Best parameters for RF:

1. max_depth: 30
2. max_features: 'auto'
3. min_samples_split: 16
4. min_samples_leaf: 2

Best parameters for XGBoost:

1. alpha: 0.1
2. lambda: 0.1
3. max_depth: 30

Because LR is my main algorithm, it will be my primary target for analysis.
I compute the standard deviations of scores associated with LR and compare with dummy classifier (baseline). The results are presented in Table 3.

	Baseline	LR avg	LR std	number of std's better
Accuracy	0.161	0.938	0.035	22.00
Precision	0.027	0.939	0.034	26.56
Recall	0.161	0.938	0.035	22.00
$f_{0.5}$	0.032	0.938	0.035	25.91
f_1	0.045	0.938	0.035	25.14
f_2	0.080	0.938	0.036	24.14

Table 3: Average score for baseline and LR. It also has LR score standard deviation and the number of std's the LR average scores are higher than the corresponding baseline scores.

I use two methods to determine global feature importance: Permutation and global SHAP. For permutation, I permute the data for each feature at a time and retrain LR. Then I select the top 10 features that have the most influences on the LR performance in each evaluation metric. The results are shown in Figure 5.

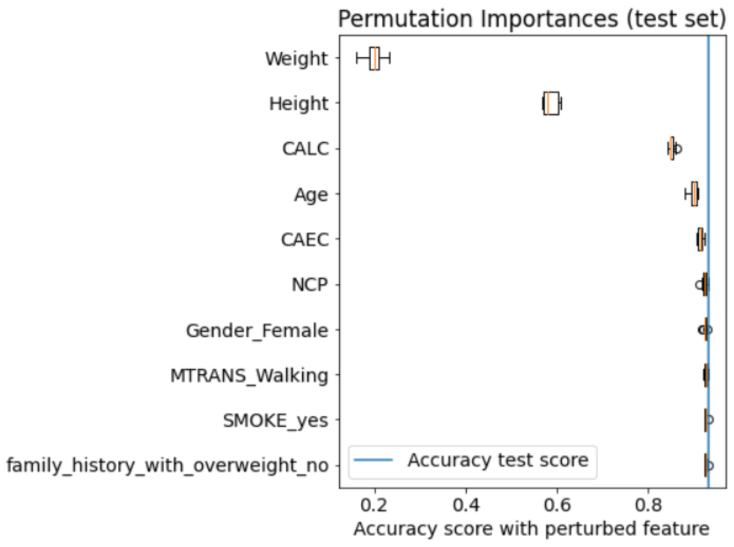

Figure 5: Top 10 features with the greatest global feature importance determined by a change of model accuracy due to perturbation.

We can see that Weight is the most important global feature. Height is not as important as Weight, but is still more significant than any other feature. The least important global feature is MTRANS_Bike, which is a binary feature that is "yes" when a person rides a bike for transportation, or "no" otherwise. The second and the third least important features are consumption of water(C2HO) and Gender_Male. They assume so little importance during the training process so that permuting them might even get slightly better metric scores due to pure luck, as shown in Figure 6.

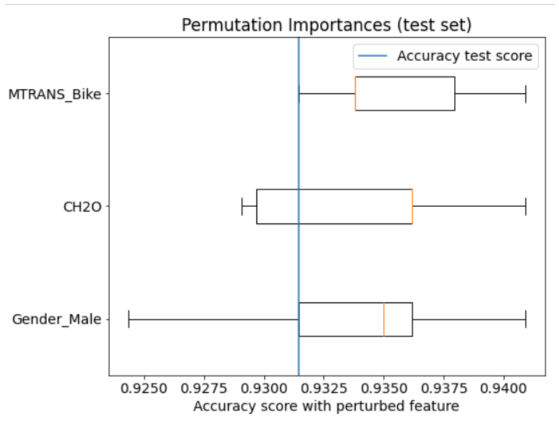

Figure 6: Top 3 features with the lowest global feature importance determined by a change of model accuracy due to perturbation.

We can also visualize the global SHAP results, Figure 7. Weight and Height remain the most important features.

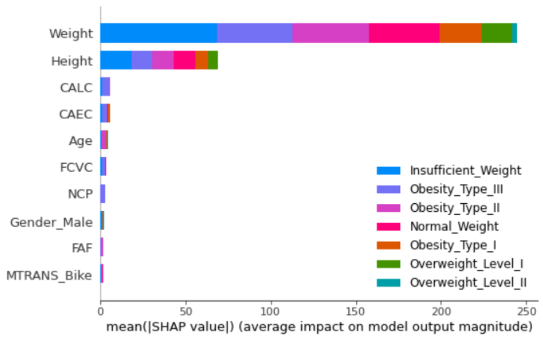

Figure 7: Global SHAP for all 7 classes in the target variable.

Because there are 7 classes, we can plot 7 force plots of local SHAP for each data point. Figure 8 is one of them.

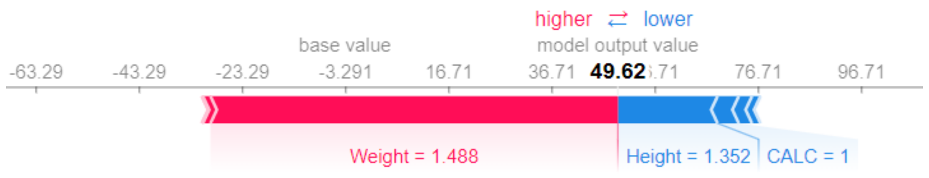

Figure 8: Local SHAP for a particular class of target variable for a person.

It is not surprising that Weight and Height are generally considered as the most important feature to determine a person's obesity. After all, they are directly used into calculating a person's Body Mass Index (BMI) which in turn determines that person's obesity level.
Perhaps the most interesting observation is that LR outperforms RF and XGBoost. This is probably due to the fact that there are only two features that dominate the feature importance ranking: Weight and Height. They are generally considered as the most direct information to determine whether a person is obese or not. Therefore, this a lot of predictive power lie within these two variables, and LR performs well probably because they have a strong correlation or anti-correlation to obesity level. If a person weighs more, he or she tends to be more obese. If a person is taller, he or she tends to be less obese. Another reason may be that LR is easy to fine-tune. I only tuned 2 LR parameters and its parameter grids are fine. It does not take a significant amount of time to train a LR due to its simplicity. Therefore, I can afford to construct a fine grid to obtain desired result. The same may not be said for RF and XGBoost. Their grids have higher dimensions, and training them takes more time due to their complexities. The tuning results may be far from optimal, and therefore they perform worse than LR.

5 Outlook

I used grid search for parameter tuning, but the grids are rather coarse, especially for RF and XGBoost. For algorithms with many parameters, tuning through a fine parameter grid usually takes a lot of time, and the results may still be sub-optimal. If time permits, I would use grid search to get a rough estimation of optimal hyperparameters, then keep fine-tuning by using gradient descent in the neighborhood of the results obtained from grid search. Aside from tuning, the process of data collection can also be improved. The data were collected from web user inputs in forms of questionnaires. People may not put in information that accurately reflect themselves. I would collect data from more authoritative sources such as patient records from hospitals. Furthermore, the data were mostly collected from college students whose age are around 20. It would be better to survey a wider age range of people as it would make the obesity study more general.

References

[1] Palechor, F. M., de la Hoz Manotas, A. (2019)., *Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico*. Data in Brief, 104344.

[2] De-La-Hoz-Correa, E., Mendoza-Palechor, F. E., De-La-Hoz-Manotas, A., Morales-Ortega, R. C. Beatriz Adriana, S. H. (2019). *Obesity Level Estimation Software based on Decision Trees*, Journal of Computer Science, 15(1), 67-77. https://doi.org/10.3844/jcssp.2019.67.77