HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

MACHINE LEARNING AND DATA MINING - IT3191E

CAPSTONE PROJECT: Obesity Level Classification

Instructor:	Assoc. Prof. Than Quang Khoat
Class:	147828
Students:	Nguyen Chi Long - 20210553
	Ngo Xuan Bach - 20215181
	Le Xuan Hieu - 20215201
	Dinh Viet Quang - 20215235
	Nguyen Viet Thuan - 20210826

Ha Noi, May - 2024

Contents

Abstract

Obesity is one of the significant public health being concerned worldwide, which can possibly lead to numerous harmful health outcomes. In our Obesity Level Classification project, we aimed to develop several classification approaches to predict obesity levels based on various demographic, lifestyle and health-related factors. Our dataset contains information from a large cohort of individuals, including characteristics, dietary, routine, habits, physical activity levels and medical history. We applied machine learning techniques to build predictive models for obesity classification. Furthermore, we performed extensive and intensive analysis on these models and also tried some approaches to reach better results. After trial and error, we finally got a 0.9153 accuracy score on the overall split test set of dataset.

1 Problem Introduction

Obesity is known to be a significant public health concern globally, contributing to numerous diseases such as cardiovascular diseases, diabetes and certain cancers. The World Health Organization (WHO) estimates that obesity has been affecting over millions of individual and posing a major challenge to healthcare systems until now. Accurate classification of obesity levels is crucial for early precautions and effective treatment.

Given the problem and the datasets, we applied variety of techniques for data analysis and preprocessing and then used well-known machine learning models to find the optimal outcome. Finally, we plotted results table to see how our model assessed the dataset.

In the report, we will outline the process we tackled the aforementioned challenge step by step. First of all, we conducted a thorough examination to define the problem definition. Subsequently, exploration data analysis is second step to discover interesting insights in our dataset. Furthermore, we built and trained some popular machine learning models. Additionally, we will detail the methodologies and techniques implemented to enhance the efficiency of each model over time, in this project, cross-validation, learning curves and other optimization strategies are used. The content in this article is referenced from [6].

2 Brief Description

The first step in any project, especially machine learning project, is correctly define the problem we need to solve in order to choose a correct strategy and suitable techniques and models. Our problem is a supervised learning problem as we need to know about the previous obesity status of a person based on his/her dietary and behaviors to make a prediction of unseen knowledge.

We define the formula that can be described as a triple *(T, E, P)*:

- Task (T): Predict the obesity level of individuals based on their health and lifestyle factors.
- Experience (E): Use a labeled dataset which is a survey list of individuals' health status and lifestyle factors.
- Performance (P): Evaluate the accuracy and other relevant metrics of the model's predictions on a test dataset to measure the difference between the predicted value and the real value.

With the aim of finding out the methodology to process data and the models applied to this situation, we tried our best to find the optimal solution to this problem.

3 Data Preprocessing

3.1 Datasets

The dataset was originally obtained from the Obesity or CVD risk Dataset [2] and Multi-Class Prediction of Obesity Risk Contest [4]. The data consist of the estimation of obesity levels in people from the countries of Mexico, Peru and Colombia, with ages between 14 and 61 and diverse eating habits and physical condition, data was collected using a web platform with a survey where anonymous users answered each question, then the information was processed obtaining 17 attributes and 22869 records in total. With 24 duplicates which is not a significant quantities, we decided to reduce our dataset to 22845 records. We split the information into the dataset, denoted by $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ containing $n = 22845$ samples and the output vector, denoted by $\mathbf{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\}$. Each input $x^{(i)}$ consists of $d = 16$ features and corresponds to each output $y^{(i)}$ which belongs to the set of classes $A = \{0, 1, \ldots, a-1\}$ in the output vector $\mathbf{y} \in A^n$. In this problem, $a = 7$ is the number of classes. The dataset has the following attributes displayed in the Table 1.

The numerical features are ‘Age’, ‘Height’, ‘Weight’, ‘FCVC’, ‘NCP’, ‘CH2O’, ‘FAF’, ‘TUE’, whereas the categorical features are ’‘Gender’, ‘family_history_with_overweight’, ‘FAVC’, ‘CAEC’, ‘SMOKE’, ‘SCC’, ‘CALC’, ‘MTRANS’. In the dataset, every attribute is fully provided. We will take a closer look at the data **in the next section**.

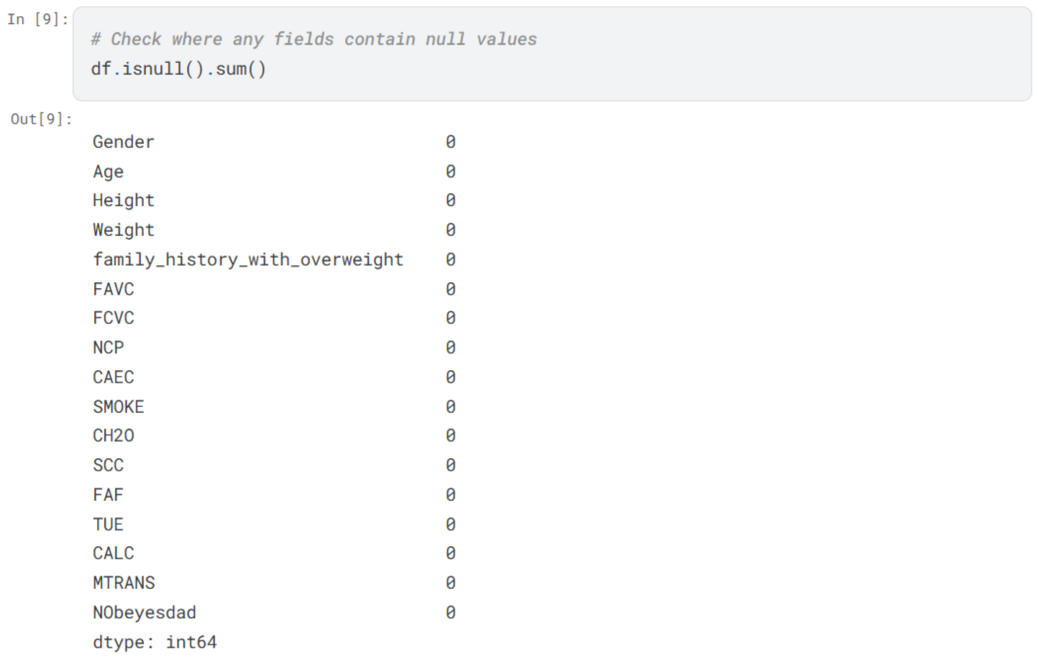
Table 1: Description of Columns for Obesity Level Prediction

Column Name	**Description**
Gender	Male or Female
Age	Years old
Height	Height (meters)
Weight	Weight (Kilograms)
family_history_with_overweight	Yes/no question, to see if person has family history with overweight
FAVC	Frequent consumption of high caloric food - yes/no question, to see if person eats high caloric food frequently
FCVC	Frequency of consumption of vegetables - Similar to FAVC but with vegetables (frequency)
NCP	Number of main meals - NCP’s value (float) ranged from 1 to 4
CAEC	Consumption of food between meals - takes 4 values (Always, Frequently, Sometimes, No)
SMOKE	Do you smoke? - yes/no question
CH2O	Consumption of water daily - CH2O’s values (float) ranged from 1 to 3
SCC	Calories consumption monitoring - Do you monitor your calories consumption? yes/no question

Column Name	Description
FAF	Physical activity frequency - ranged from 0 to 3 (float), where 0 means no physical activity and 3 means frequent activity
TUE	Time using technological devices such as smartphone or laptop - ranged from 0 to 2, where 0 means no devices used while 2 means high frequency of devices used.
CALC	Consumption of alcohol - takes 4 values: Always, Frequently, Sometimes, No
MTRANS	Transportation used - takes 5 values: Public Transportation, Automobile, Walking, Motorbike, & Bike
NObeyesdad	This is our target value, takes 7 values

3.2 Exploratory Data Analysis

First of all, we have to check the data reliability if any data in a row is missing in the dataset. In Figure 1, we can see that there are no missing data across all columns. Therefore, we do not need any imputation strategies or data manipulation techniques, which may introduce bias or reduce the accuracy of predictive models.



```
In [9]:  # Check where any fields contain null values
         df.isnull().sum()

Out[9]:
          Gender                            0
          Age                               0
          Height                            0
          Weight                            0
          family_history_with_overweight    0
          FAVC                              0
          FCVC                              0
          NCP                               0
          CAEC                              0
          SMOKE                             0
          CH2O                              0
          SCC                               0
          FAF                               0
          TUE                               0
          CALC                              0
          MTRANS                            0
          NObeyesdad                        0
          dtype: int64
```

Figure 1: The number of missing data

Next, we want to analyse the dataset relating to the categorical and numerical column in Figure 2 and Figure 3. Our dataset contains several categorical columns that provide valuable insights into the characteristics and behaviors of the individuals. The gender distribution in the dataset is fairly equal, ensuring the balanced representation. Many individuals report a family history of being overweight. Additionally, the same trend occurs for the participants consume high-caloric foods frequently, which is a concerning factor in obesity risk. Furthermore, the majority of individuals sometimes consume food between meals, while a few report not eating food in meals. The dataset also reveals that almost all of the people are non-smokers, as well as do not monitor their calorie consumption. Alcohol consumption is also worth noting, with large percentage of individuals sometimes consuming alcohol. Furthermore, nearly all participants use some kind of vehicle for transportation, indicating a low level of physical activity related to traveling.

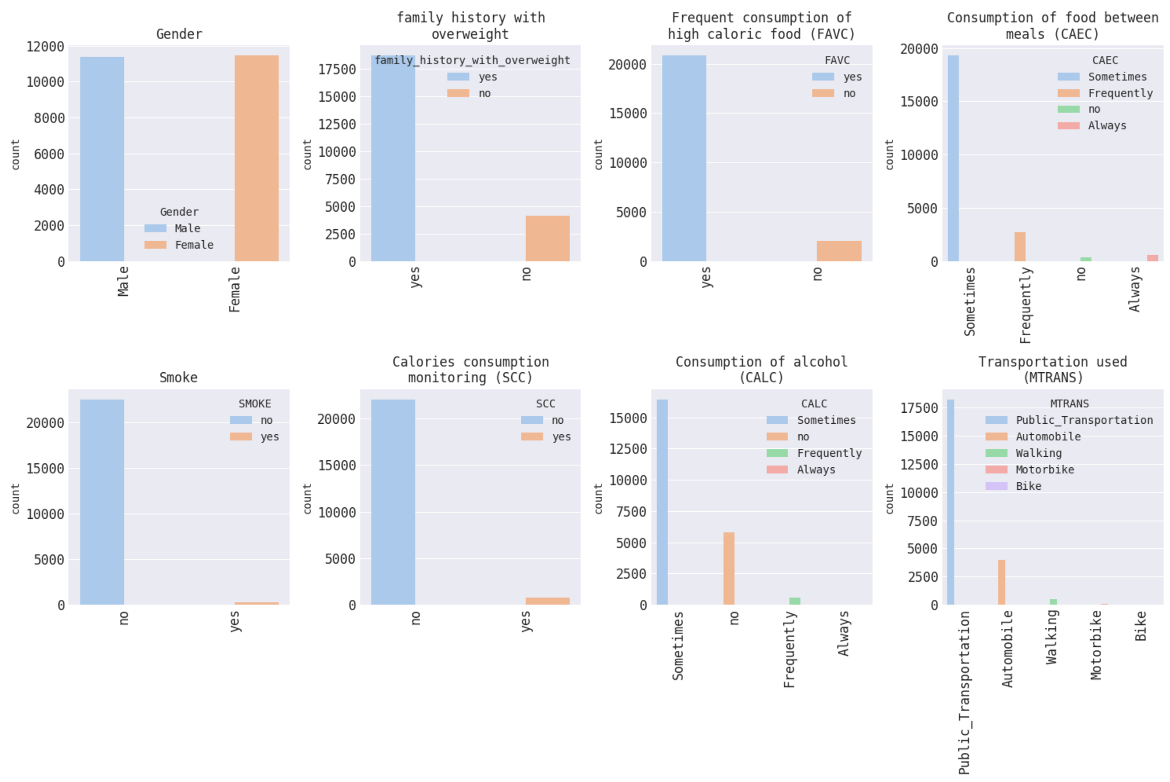


Figure 2: The data distribution of categorical columns

Numerical columns' data distrubution in Figure 3 will be discussed below. First of all, the age column represents the age of individuals. Its distribution typically follows a skewed pattern, with most people surveyed belongs to an age range from 20 to 30. However, there might be outliers (higher ages) that could impact predictive models. Secondly, the height and weight distribution are quite close to a normal distribution. On the one hand, FCVC shows how often individuals consume vegetables. Its distribution is skewed, with most people tend to high vegetable consumption trend. On the other hand, NCP indicates the number of main meals consumed daily. The distribution might be bimodal with two peaks, with some people having two or three meals a day and others having more or fewer. CH2O represents daily water intake. The distribution could be right-skewed, with most people consuming an average amount of water. In addition, FAF reflects how often individuals engage in physical activity. The distribution shows that people moderate activity levels (1 - 2 points) and people with no physical activity (0 points) are majority. Last but not least, TUE measures behavior related to technological devices use. We can see that most people spending some of their time using devices, ranged from 0 to 2.

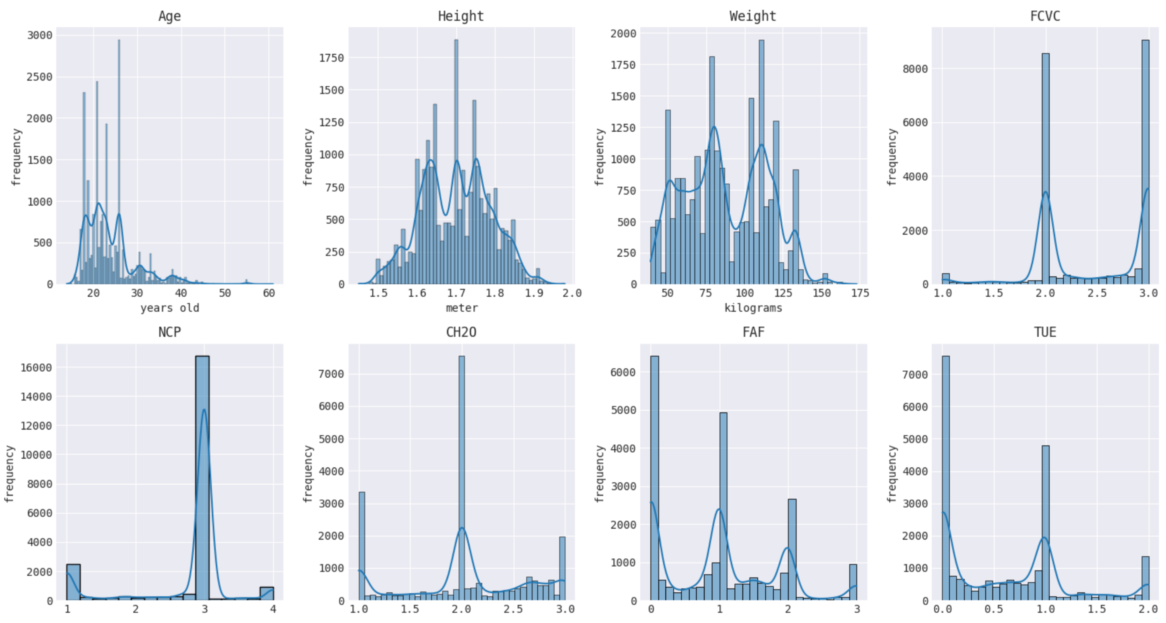


Figure 3: The data distribution of numerical columns

We consider the label distribution in our dataset to check the balance of it. This distribution provides insights into the prevalence of various obesity levels within the dataset. The Figure 4 demonstrates a notable balance among various weight categories, with Obesity_Type_III being the most prevalent category representing 19.1% of the dataset. In contrast, the least common label with 11.8% is Overweight_Level_I. Overall, we can see that the number of records for each label is pretty balanced. The balanced distribution of labels in the dataset significantly enhances the model training process for reducing bias and overfitting phenomenon.

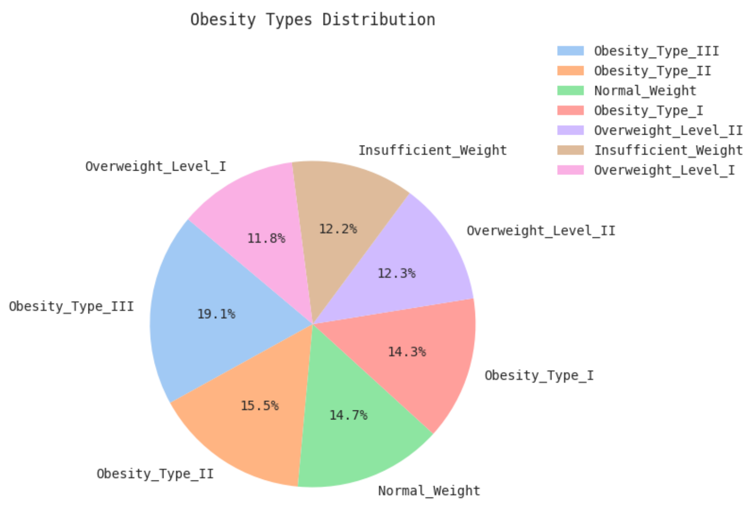


Figure 4: The NObeyesdad label distribution of the dataset

In Figure 5, we plot the label distribution relating to the Gender to see if there is a connection between them. In general, this figure reveals some distinct patterns. There are more females with insufficient weight compared to males, indicating a higher prevalence of underweight issues among women. In contrast, almost all respondents classified as Obesity_Type_II are male, while most individuals classified as Obesity_Type_III are female.

This gender disparity highlights significant differences in obesity severity between men and women. Notably, more than 4,000 female respondents fall into the Obesity_Type_III category, making a critical concern for women's health within this population.

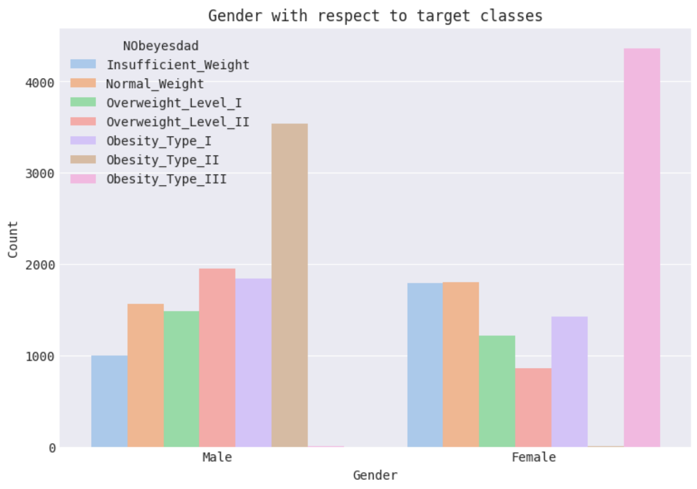


Figure 5: NObeyesdad label distribution with respect to Gender

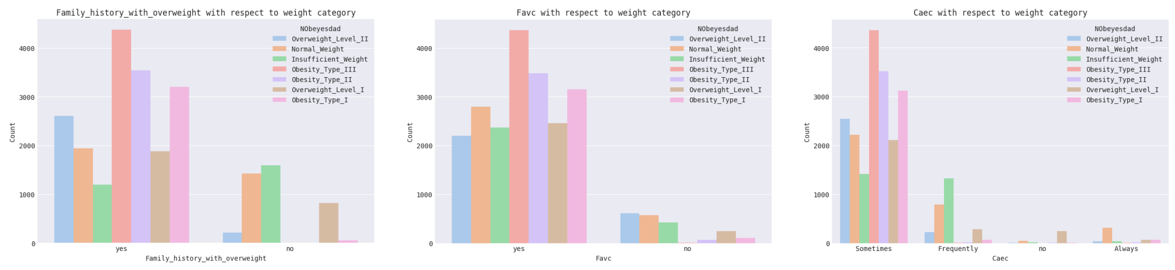


Figure 6: NObeyesdad label distribution with respect to family history status, FAVC and CAEC

Next, we would like to dive deeper into the relationship between each categorical column and NObeyesdad. The first bar chart in Figure 6 illustrates the distribution of various weight categories among individuals with a family history of overweight. Individuals with no family history show a higher count in the Normal_Weight and Insufficient_Weight category. In contrast, Obesity_Type_I is notably prevalent among those with a family history. Obesity_Types_II and III observes a moderate counts regardless of family history.

The second figure compares the number of individuals across different weight categories based on their frequency of high caloric food consumption (FAVC). Those who frequently consume high caloric food ('yes') show a significant count in the Obesity_Type_I, II and II category, suggesting a correlation between diet and obesity. Individuals who do not frequently consume high caloric food ('no') predominantly fall within the Normal_Weight category, indicating a possible link between lower caloric intake and normal weight.

When it comes to CAEC (Consumption of food between meals), it is essential to shows the relationship between the frequency of eating between meals and various weight categories. Individuals who do not consume food between meals ('no') or do it 'Sometimes' have the highest counts in the Normal Weight category. Additionally, Overweight_Level_I is more popular among the labels for those who 'frequenlty' eat food between meals. On the other hand, the 'no' and 'Always' category shows a minimal counts across all NObeyesdad categories. This distribution suggests that most of individual sometimes eating food.

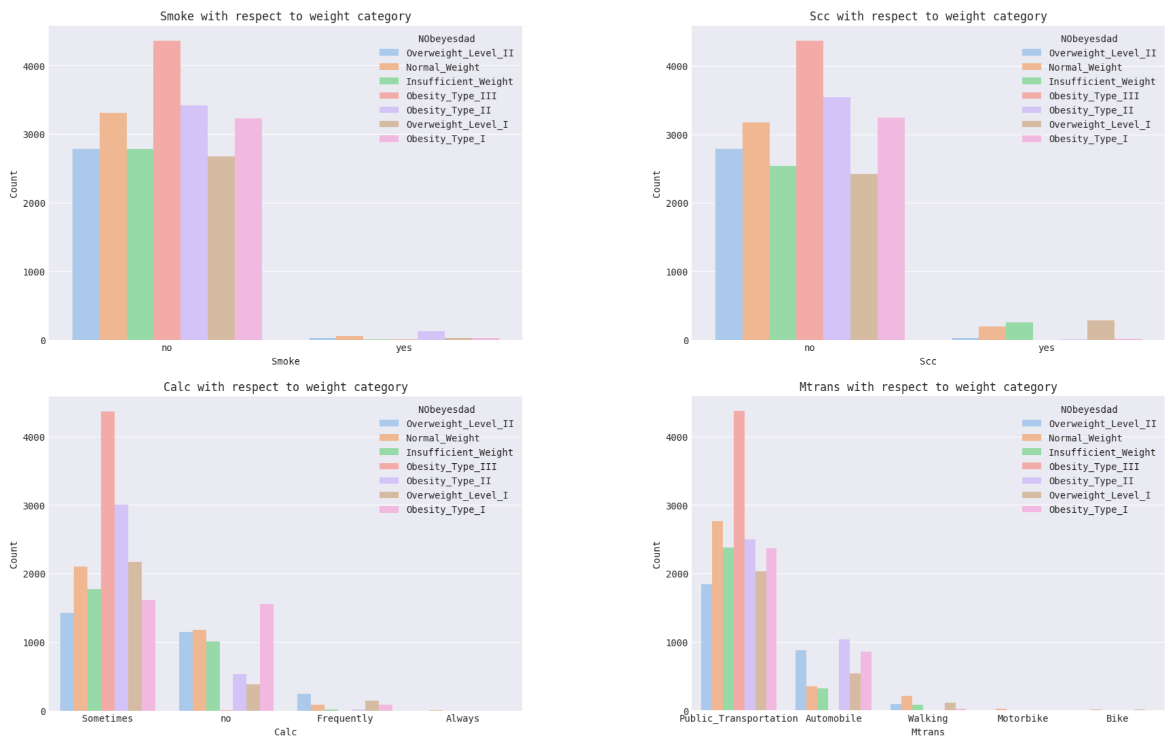


Figure 7: NObeyesdad label distribution with respect to SMOKE, SCC, CALC and MTRANS

The following Figure 7 represent the insights into the daily habits and routines of individual with different body weight classifications, namely SMOKE, SCC (Calories consumption monitoring), CALC (Consumption of alcohol) and MTRANS (Transportation used). In SMOKE bar chart, A higher count of non-smokers is observed across all weight categories, with the Normal_Weight category having the most number of records. In contrast, the number of smokers is significantly lower in comparison to non-smokers in each weight category. The SCC figure provides insights into the distribution of weight categories among individuals based on their self-monitoring of calorie consumption, where individuals who do not monitor their calorie intake show a diverse distribution across weight categories, with the highest record counts for Obesity_Type_III, as expected. Those with lower level of obesity tend to have dietary monitoring, although this counts is significantly lower than the other one, suggesting a possible unawareness regarding their weight status. For the relationship between alcohol consumption frequency and various weight categories, individuals who consume alcohol ‘Sometimes’ or ‘no’ show a majority record, where the highest counts belongs to the Normal_Weight category. On the other hand, A relatively small count across all weight categories is observed for those who ‘always’ or ‘frequently’ consume alcohol. The third figure (MTRANS) illustrates the distribution of various weight categories among different kinds of transportation. Public Transportation mostly used by individuals across all weight categories, with the majority being in the Normal Weight category. Followed by the second most common transportation vehicle is Automobile, with significant counts in Normal Weight, Overweight Level I and Obesity Type I categories. However, Active transport such as Walking and Bike shows the least common records.

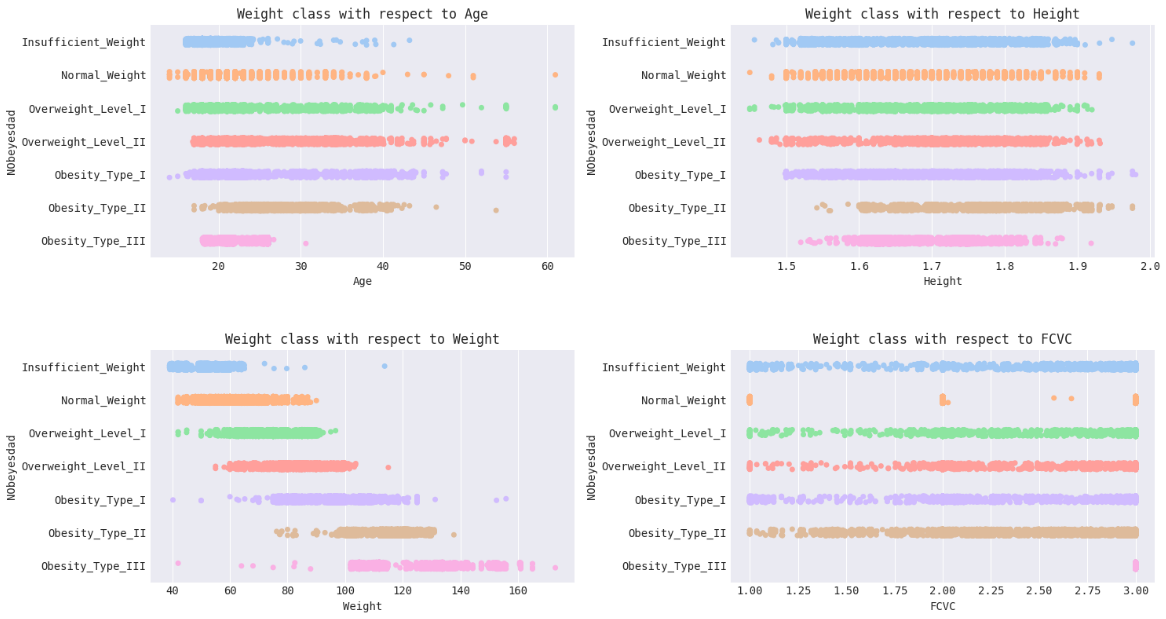


Figure 8: The first scatter plot

Figure 8 provides insights into body measurements and dietary intake across different body weight classifications, focusing on Age, Height, Weight, and FCVC (Frequency of Consumption of Vegetables).

In the Age scatter plot, higher obesity types (Level I, Level II, Type I, and Type II) become more prevalent as individuals age, peaking in the 30s and 40s. Younger individuals, especially those under 20, are more likely to fall into the Insufficient Weight and Normal Weight categories. Interestingly, Type III Obesity is seen only in the 20 to 26-year-old age group. This pattern indicates that weight issues, particularly severe obesity, tend to become more prominent with age.

The Height plot reveals that individuals shorter than 1.5 meters are not classified under Type II and Type III obesity categories. Those with heights between 1.6 and 1.8 meters are more likely to suffer from severe obesity levels, including Level I, Level II, Type I, Type II, and Type III obesity. This suggests a significant relationship between height and obesity classification, with shorter individuals being less represented in the severe obesity categories.

The Weight plot illustrates a step-like diagram, showing that the severity of obesity increases progressively with weight. This correlation is logical, as higher weights correspond to more severe obesity classifications, highlighting a direct relationship between weight and obesity severity.

Lastly, the FCVC plot shows a wide range of vegetable consumption frequencies among individuals in all weight classes, from Insufficient Weight to Obesity Type III. There is no clear trend linking vegetable consumption frequency directly to weight class, suggesting that how often individuals consume vegetables does not significantly influence their weight classification. This indicates that other factors play a more substantial role in determining an individual's weight class. Overall, Figure 8 illustrates that while age, height, and weight show clear correlations with obesity levels, the frequency of vegetable consumption alone is not a significant determinant of weight classification. This underscores the complexity of factors contributing to obesity and weight management.

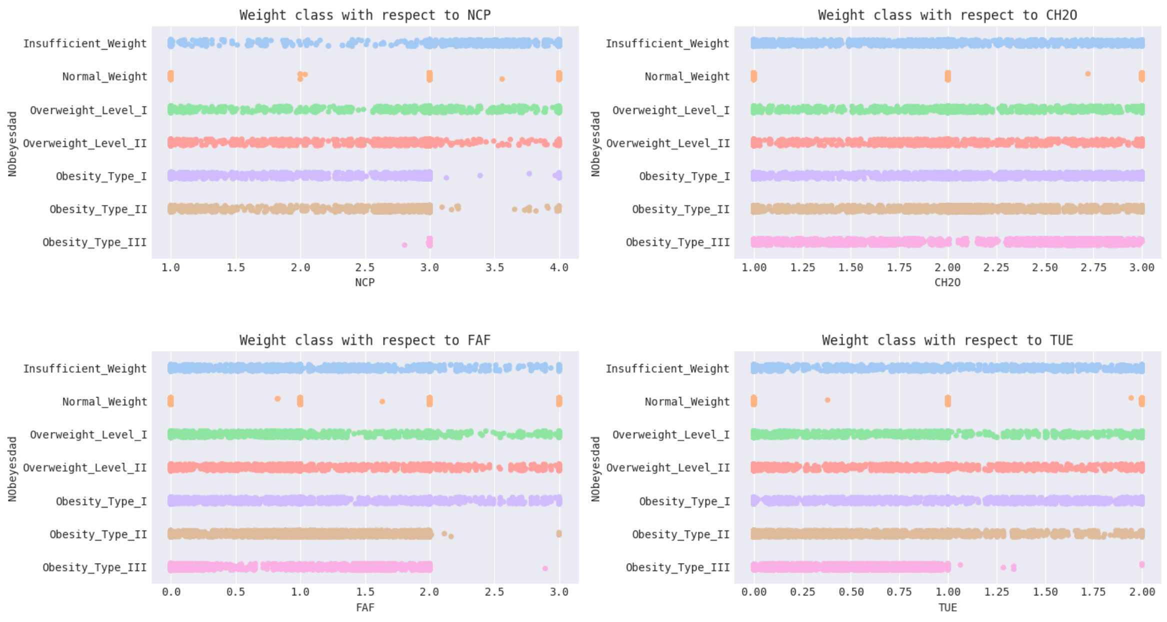


Figure 9: The second scatter plot

Figure 11 provides insights into the daily activities and dietary intake of individuals across different body weight classifications, focusing on NCP (Number of Main Meals), CH2O (Daily Water Consumption), FAF (Physical Activity Frequency), and TUE (Time Using Technological Devices).

The NCP plot illustrates that individuals in all weight classes consume between 1 to 4 main meals per day. Noticeably, a significant clustering is observed around 3 main meals in Type III Obesity. However, there is no clear trend showing that the number of main meals significantly distinguish between different weight classes.

The CH2O plot shows that individuals in all weight classes consume varying levels of water daily. There is a noticeable clustering around the 2 value mark for most weight categories. This indicates that daily water intake habits are quite similar across different body weight classifications, with the majority of individuals consuming approximately the same level of water each day . However, there is no clear trend showing that higher or lower water consumption is associated with any particular weight class, suggesting that water intake does not significantly determine weight class.

The FAF plot shows that individuals across all weight classes engage in varying levels of physical activity, with most data points clustered around lower FAF values. This suggests that lower physical activity levels are common regardless of weight class, and higher FAF values are less frequent, indicating generally low levels of physical activity among the population. However, it is interesting that those who frequently exercise (2.0 to 3.0) fall into the Overweight Level I, II and Obesity Type I which seems paradoxical. This pattern indicates that physical activity frequency does not significantly differentiate between weight classes.

The TUE plot demonstrates that individuals across all weight categories generally have TUE values clustered around the lower end. This suggests that most people spend relatively less time using technological devices, regardless of their weight class. The spread of TUE is uniform across weight classes, indicating that technology use does not vary significantly with weight, and thus, does not play a major role in determining weight class. Overall, Figure 11 highlights that the number of main meals, daily water consumption, and time spent using technological devices are similar across different weight classes. In contrast, lower levels of physical activity, typically around one or fewer sessions per week, are prevalent regardless of weight. This suggests that factors other than meal frequency, water intake, and technology use might play more substantial roles in determining weight classifications.

The relationship between height, weight and NObeyesdad in the dataset shows several important insights in Figure 10. First of all, a positive correlation between height and weight, indicating that taller individuals tend to be heavier and vice versa. When examining the obesity classifications, individuals labeled with 'Obesity_Type' categories generally have higher weights compared to those with lower weight categories. Furthermore, there seems to be a cluster of individuals with 'Obesity_Type_III' who have shorter heights and heavier weights, indicates a group of severely obese individuals who are much heavier for their height, showing the complexity and differences in obesity classifications.

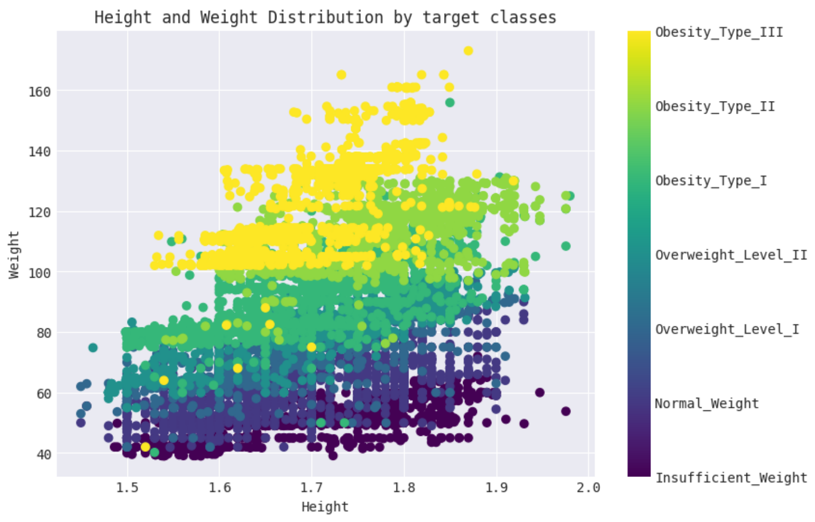


Figure 10: The relationship of height and weight, colored by NObeyesdad label

3.3 Preprocessing Method

Splitting Features and Encoding Target Classes: The target variable `NObeyesdad` was separated from the features and were encoded into numerical values using a predefined mapping to facilitate machine learning algorithms.

Transforming Skewed Features: The `Age` column was transformed to approximate a normal distribution using the Box-Cox transformation.

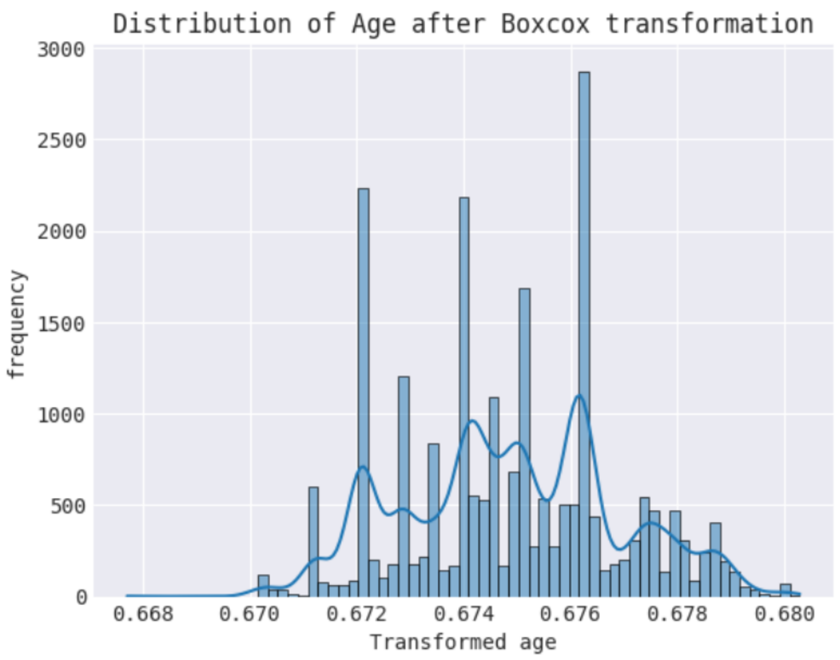


Figure 11: Distribution of Age after Boxcox transformation

Discretizing Continuous Features: Several columns (`FCVC`, `NCP`, `CH2O`, `FAF`, `TUE`) were transformed into discrete values. These features had values that naturally clustered around certain points. To simplify and standardize the data, these continuous values were rounded up to the nearest discrete categories using the `pd.cut` method. This transformation helps in reducing variability and enhancing the interpretability of these features.

Column Transformation: A `ColumnTransformer` was set up to preprocess the data. This included one-hot encoding for certain categorical variables, ordinal encoding for others, and standard scaling for numerical variables.

```
ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore
    ', drop='if_binary')
oe = OrdinalEncoder(categories=[['no', 'Sometimes', 'Frequently
    ', 'Always']] * 2)
scaler = StandardScaler()
col_trans = make_column_transformer(
    (ohe, ['Gender', 'family_history_with_overweight', 'FAVC',
        'SMOKE', 'SCC', 'MTRANS']),
    (oe, ['CAEC', 'CALC']),
    (scaler, ['Age', 'Height', 'Weight']),
    remainder='passthrough',
    n_jobs=-1
)
col_trans.set_output(transform='pandas')
```

3.4 Feature Engineering

After conducting exploratory data analysis (EDA), it was evident that weight and height significantly influence the model's performance. BMI, calculated using weight and height, emerged as a potentially important feature. Therefore, polynomial features based on BMI, weight and height were introduced to capture more complex relationships.

4 Machine Learning models

In this section, we will introduce the theories of each model and our reason for selecting each of them to tackle this challenging problem. We will begin with K-Nearest Neighbor, a baseline model, to different models such as Decision Trees and Logistic Regression. Finally, Ensemble Learning such as XGBoost is discussed.

4.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) implements learning based on the nearest neighbors of each query point, where K is an integer parameter. Briefly, KNN is an algorithm that finds the output of a new data point by only basing on the information of the K data points in the training set that are closest to it (K-neighbors), regardless of noisy data points. Here, the optimal value of K is highly data-dependent: in general a larger suppresses the effects of noise, but makes the classification boundaries less distinct.

To find these neighbors, we use the Minkowski distance with parameter p. For 2 points $x^{(i)}$ and $x^{(j)}$, the distance is computed as: $\|x^{(i)} - x^{(j)}\|_p = \left(\sum_{k=1}^{d} |x_k^{(i)} - x_k^{(j)}|^p\right)^{1/p}$.

We use uniform weights: that is, the class assigned to a query point is the class that appears most frequently in the K nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit.

The predicted output $\hat{y} \in A$ for an input x is:

$$\hat{y} = \arg\max_{k \in A} \sum_{i \in \{i_1, i_2, \ldots, i_k\}} \mathbb{I}(y^{(i)} = k) \tag{1}$$

where:

- $\{i_1, i_2, \ldots, i_K\}$ is the set of indices of K nearest neighbors of x in the train set.
- $\mathbb{I}(y^{(i)} = k)$ is an indicator function that equals 1 if $y^{(i)} = k$ and 0 otherwise.

4.2 Multinomial Logistic Regression

Let $y^{(i)} \in A$ be the target variable for observation i. We now have a matrix of coefficients W where each row vector W_k corresponds to class k. We can predict the class probabilities $P(y^{(i)} = k | x^{(i)})$ by softmax function as:

$$P(y^{(i)} = k|x^{(i)}) = \hat{p}_k(x^{(i)}) = \frac{\exp(W_{1:d,k} x^{(i)} + W_{0,k})}{\sum_{l=1}^{K} \exp(W_{1:d,l} x^{(i)} + W_{0,l})}$$

The objective for the optimization is:

$$\min_W -C \sum_{i=1}^{n} \sum_{k=0}^{a-1} [y^{(i)} = k] \log(\hat{p}_k(x^{(i)})) + r(W)$$

which is the cross-entropy loss, where:

- C: the inverse of regularization parameter.
- $[P]$: the Iverson bracket which evaluates to 0 if P is false, otherwise it evaluates to 1.
- $r(W)$: the regularization term.

We use the solution W^* from the algorithm used for the optimization problem and the predicted output $\hat{y} \in A$ for an input x is:

$$\hat{y} = \arg\max_{k \in A} \hat{p}_k(x) = \arg\max_{k \in A} \frac{\exp(W^*_{1:d,k}x + W^*_{0,k})}{\sum_{l=0}^{a-1} \exp(W^*_{1:d,l}x + W^*_{0,l})} \tag{2}$$

4.3 Decision Tree

Decision trees are highly effective for both regression and classification tasks. They employ a series of if-else rules, which makes them particularly adept at handling datasets composed entirely of categorical features. Moreover, decision trees form the foundation for many advanced decision-making and ensemble learning techniques.

A decision tree is structured as a tree, where each path from the root to a leaf node signifies a distinct decision. Each branch in the tree reflects the values of the attributes at the node from which the branch originates.

Given training input data $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\} \in \mathbb{R}^{n \times d}$ and a label vector $\mathbf{y} \in A^n$, a decision tree recursively partitions the feature space so that samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by X_m with n_m samples. For each candidate split $\theta = (j, t_m)$, consisting of a feature j and threshold value t_m, the data is partitioned into $X_m^{\text{left}}(\theta)$ and $X_m^{\text{right}}(\theta)$ subsets:

$$X_m^{\text{left}}(\theta) = \{x \in X_m \mid x_j \leq t_m\}$$
$$X_m^{\text{right}}(\theta) = X_m \setminus X_m^{\text{left}}(\theta)$$

Let n_m^{left} and n_m^{right} be the size of $X_m^{\text{left}}(\theta)$ and $X_m^{\text{right}}(\theta)$, respectively. Note that: $n_m^{\text{left}} + n_m^{\text{right}} = n_m$. The quality $G(X_m, \theta)$ of a candidate split at node m by using θ is evaluated using an impurity function or loss function $H(\cdot)$:

$$G(X_m, \theta) = \frac{n_m^{\text{left}}}{n_m} H(X_m^{\text{left}}(\theta)) + \frac{n_m^{\text{right}}}{n_m} H(X_m^{\text{right}}(\theta))$$

The split θ^* that minimizes the quality is chosen:

$$\theta^* = \arg\min_{\theta} G(X_m, \theta)$$

This process recurses for subsets $X_m^{\text{left}}(\theta^*)$ and $X_m^{\text{right}}(\theta^*)$ until the maximum allowable depth (`max_depth`) is reached, or the number of samples at a node is less than a specified minimum, $n_m \leq$ `min_samples_leaf`. When a test case is introduced into the decision tree at the root node, it undergoes a series of conditional evaluations to determine the subsequent node at each step. This process continues until the test case reaches a leaf node, also known as a terminal node.

For classification tasks where the target takes on values $k \in A = \{0, 1, \ldots, a-1\}$, let

$$p_{mk} = \frac{1}{n_m} \sum_{i:\, x^{(i)} \in X_m} \mathbb{I}(y^{(i)} = k)$$

be the proportion of class k observations in node m. Common measures of impurity include:

Gini:

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy:

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$$

The predicted output $\hat{y} \in A$ for an input x is:

$$\hat{y} = \arg\max_{k \in A} p_{mk} \tag{3}$$

where m is the terminal node.

4.4 Random Forest

A Random Forest is an ensemble learning method designed to enhance predictive accuracy and robustness by combining multiple decision trees. Each tree in the forest is built using a bootstrap sample of the training data, where samples are drawn with replacement. This introduces variability in the training set for each tree, ensuring that the trees are diverse. When constructing each tree, the algorithm also considers random subsets of features at each split, further promoting diversity among the trees.

This randomness in both the sampling of data and the selection of features helps to reduce the variance typically associated with individual decision trees, which tend to overfit their training data. By averaging the predictions of the numerous decision trees, the Random Forest effectively reduces overfitting and variance. While this may introduce a slight increase in bias, the overall effect is a more accurate and generalizable model. The combination of diverse, low-correlation trees results in a powerful predictor that usually performs better than any individual tree in the ensemble.

In the algorithm, we will use T decision trees. We use the same weight for every class, so the predicted output $\hat{y} \in A$ for an input x is:

$$\hat{y} = \arg\max_{k \in A} \sum_{j=1}^{T} p_{jk} \tag{4}$$

where p_{jk} is prediction probability for label k of j^{th} tree.

4.5 XGBoost

XGBoost (Extreme Gradient Boosting), first introduced in [3], is an advanced implementation of gradient boosting that has become widely popular for its efficiency and high performance in machine learning competitions.

4.5.1 Boosting

Boosting is a supervised machine learning strategy that combines the predictions of multiple weak models (a model that performs slightly better than random guessing) to generate a powerful ensemble model. It focuses on successively training the basic models in a way that emphasizes misclassified samples from prior iterations, so the model can learn from its mistakes and improve its performance iteratively.

In the boosting algorithm, we start with a base learner, which reads the data and assigns equal weight to each sample observation. False predictions made by the base learner are identified, and the predictions are assigned to the next base learner such that the incorrect predictions are emphasized. And we repeat this until triggering the stopping condition.

4.5.2 Gradient Boosting

In gradient boosting, each tree is trained to minimize the residual errors (which is the difference between the actual value and the predicted value from the current model) of the previous trees. In the training process, we apply a factor (called the learning rate) to the predictions of the weak learner before adding them to the current model. This helps prevent overfitting by controlling the contribution of each weak learner.

4.5.3 Extreme Gradient Boosting (XGBoost)

XGBoost is a regularized form of the existing gradient-boosting algorithm. Due to this, XGBoost performs better than a normal gradient boosting algorithm, and it also performs better when there is a presence of numerical and categorical features in the dataset. In the algorithm, we will use T decision trees.

During training, we need to minimize the objective function of XGBoost, which is designed to optimize both the loss function and the regularization term:

$$\text{obj} = \sum_{i=1}^{n} l(y^{(i)}, \hat{y}^{(i)}) + \sum_{j=1}^{T} \omega(f^{(j)}) \tag{5}$$

where:

- $l(y^{(i)}, \hat{y}^{(i)})$ is the loss function that measures the difference between the predicted value $\hat{y}^{(i)}$ and the actual value $y^{(i)}$ of an input $x^{(i)}$. In the case of multi-class classification, l is the cross-entropy loss function where the probability vector is calculated by softmax function, the same as in Section 4.2.

- $\omega(f^{(j)}) = \gamma L + \frac{1}{2}\lambda \sum_{k=1}^{L}(w_k^{(j)})^2$ is the regularization term for the complexity of the decision tree $f^{(j)}$, where L is the number of leaves in $f^{(j)}$, γ and λ are regularization parameters and $w_k^{(j)}$ is the score on leaf k of $f^{(j)}$. This can be considered as a combination of two regularization methods: Ridge and Lasso.

Additive Training We need to learn the decision trees $\{f^{(j)}\}_{j=1}^{T}$, each containing the structure of the tree and the leaf scores. Learning tree structure is much harder than traditional optimization problem where you can simply take the gradient. It is intractable to learn all the trees at once. Instead, we use an additive strategy: fix what we have learned, and add one new tree at a time. We write the predicted value at step t of input $x^{(i)}$ as $\hat{y}_t^{(i)}$. Then we have:

$$\begin{aligned}
\hat{y}_0^{(i)} &= 0\\
\hat{y}_1^{(i)} &= \hat{y}_0^{(i)} + \eta f^{(1)}(x^{(i)})\\
\hat{y}_2^{(i)} &= \hat{y}_1^{(i)} + \eta f^{(2)}(x^{(i)})\\
&\dots\\
\hat{y}_t^{(i)} &= \hat{y}_{t-1}^{(i)} + \eta f^{(t)}(x^{(i)})
\end{aligned}$$

where η is the step size shrinkage used in update to prevents overfitting. Assume that the values from previous steps are constant, the objective at step t becomes:

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y^{(i)}, \hat{y}_t^{(i)}) + \sum_{j=1}^{t} \omega(f^{(j)})\\
&= \sum_{i=1}^{n} l(y^{(i)}, \hat{y}_{t-1}^{(i)} + f^{(t)}(x^{(i)})) + \omega(f^{(t)}) + \text{const}
\end{aligned}$$

There are many types of loss function whose gradients are hard to calculate like Hinge loss (the derivative is a piece-wise function), so in the general case, we take the Taylor expansion of the loss function up to the second order. To recall, the second order Taylor expansion of a function f is:

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)(\Delta x)^2$$

The objective at step t becomes:

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} [l(y^{(i)}, \hat{y}_{t-1}^{(i)}) + g^{(i)} f^{(t)}(x^{(i)}) + \frac{1}{2}h^{(i)}(f^{(t)}(x^{(i)}))^2] + \omega(f^{(t)}) + \text{const}\\
&= \sum_{i=1}^{n} [g^{(i)} f^{(t)}(x^{(i)}) + \frac{1}{2}h^{(i)}(f^{(t)}(x^{(i)}))^2] + \omega(f^{(t)}) + \text{const}
\end{aligned}$$

Here:

- $g^{(i)} = \partial_{\hat{y}_{t-1}^{(i)}} l(y^{(i)}, \hat{y}_{t-1}^{(i)})$ is the gradient
- $h^{(i)} = \partial^2_{\hat{y}_{t-1}^{(i)}} l(y^{(i)}, \hat{y}_{t-1}^{(i)})$ is the Hessian

Now we need to find trees that minimize this objective, and we use the tree construction algorithm used in LightGBM [5]. In multi-label classification, for each label, we will build a XGBoost model for it, and the probability of each label in the prediction will be calculated by the softmax function.

4.6 Voting Classifier

We already have a lot of good classifiers in the previous subsections, why don't we think about combining them? Ensemble learning is a good method for the task, and in this subsection, we will talk about Voting Classifier, a model that supports this task. To combine classifiers, we need to come up with a way to produce aggregated results based on the evaluation of each classifier. For Voting Classifier, we use soft voting method, which means that the result of Voting Classifier will be the label with the highest probability based on the sum of the probabilities of the classifiers. Assume that we have T classifiers, and the predicted probability of these classifiers given input x are $\mathbf{p}_1 = [p_{1,0}, p_{1,1}, \ldots, p_{1,a-1}], \mathbf{p}_2, \ldots, \mathbf{p}_T$, where $p_{j,k}$ is the predicted probability for label k of classifier j given input x, then the predicted output $\hat{y} \in A$ for an input x is:

$$\hat{y} = \arg\max_{k \in A} \sum_{j=1}^{T} p_{j,k} \tag{6}$$

5 Experiments

After our explanation about data visualization and analysis and the theoretical properties of learning models we chose to predict the target variable, this section will describe how we applied theories with previous sections into practice. Starting from the training strategy in Subsection 5.1, where we describe how to prepare the data and which techniques we tried to improve our results. After that, we chose suitable evaluation metrics to evaluate the performances of this classification problem. Then we will describe how we trained and assessed our models using 5-fold cross validation and stratified sampling. Finally, in Subsection 5.3 and 5.4, we will present our results in quantitative matters. Bonus for our project is website demonstration using Gradio in Subsection 5.5.

5.1 Main workflow

Our classification workflow is broken down into following steps [1]:

1. Data analysis
2. Data preprocessing
3. Model selection
4. Model training
5. Evaluation
6. Model deployment

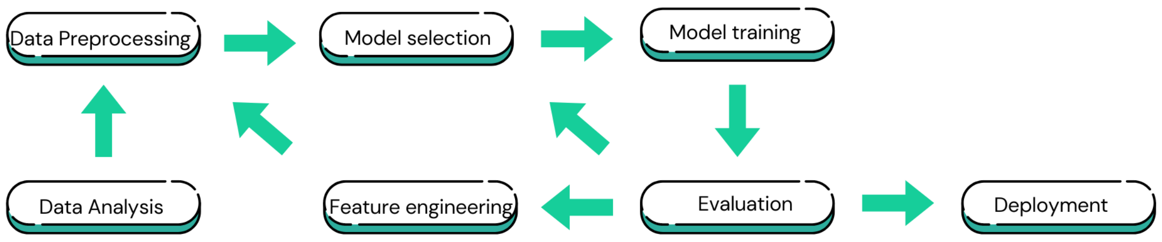


Figure 12: Main Workflow

5.2 Evaluation metrics selection

We chose accuracy as our evaluation metric for the problem because of its popularity for classification problems, which is calculated as the total number of correct classifications divided by the total number of instances. The confusion matrix, used for checking the best model after all, is also chosen as it enables us to see which classes are misclassified with which classes.

5.3 Model assessment

The dataset is divided into 80% training set and 20% test set, using **stratified sampling** to maintain the distribution of NObeyesdad in both sets. This ensured that our models were trained and evaluated on representative data. After that, we use **Stratified K-fold cross validation with $K = 5$ using RandomSearchCV** to find the model with best performance on the validation set.

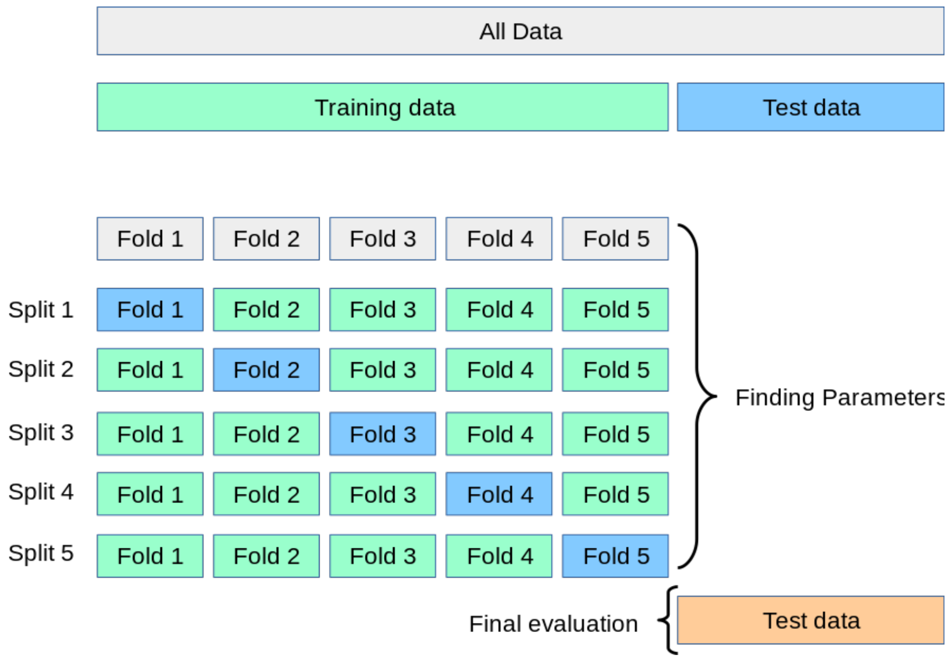


Figure 13: 5-fold cross validation

KNN: In the setup of the KNN, we tune the following parameters:

- `n_neighbors` (K): `[2, 5, 8, 10, 20]`

- p (parameter of Minkowski distance): `[1, 2]`

and here is the results:

Table 2: Results of KNN using 5-fold cross validation

params	**mean fit time**	**mean score time**	**mean test score**	**mean train score**
{'p': 1, 'n_neighbors': 20}	0.0117	3.4113	0.8297	0.8484
{'p': 1, 'n_neighbors': 10}	0.0115	3.3076	0.8280	0.8627
{'p': 1, 'n_neighbors': 8}	0.0114	3.3052	0.8272	0.8660
{'p': 1, 'n_neighbors': 5}	0.0127	3.3420	0.8193	0.8773
{'p': 2, 'n_neighbors': 10}	0.0114	0.8605	0.8025	0.8390
{'p': 2, 'n_neighbors': 8}	0.0137	0.9487	0.7998	0.8466
{'p': 2, 'n_neighbors': 20}	0.0116	0.8811	0.7968	0.8197
{'p': 2, 'n_neighbors': 5}	0.0117	0.8607	0.7956	0.8613
{'p': 1, 'n_neighbors': 2}	0.0161	3.4240	0.7790	0.8966
{'p': 2, 'n_neighbors': 2}	0.0113	0.8641	0.7642	0.8891

In our k-NN experiments, using the Manhattan distance (p = 1) significantly increased computation time compared to the Euclidean distance (p = 2), likely due to the complexity of absolute value calculations. Additionally, increasing the number of neighbors improved the model performance. Also, the nearly zero training time in k-NN occurs because the algorithm is non-parametric and instance-based, meaning it does not involve a training phase. However, the prediction time is considerably longer since it requires computing distances between the query point and all training points

Logistic Regression In the setup of Logistic Regression, we tune the following parameters:

- `max_iter`: `10000`
- `C` (C): `[1e-1, 1, 10, 100, 1000]`

Note that we also run a Logistic Regression model without regularization (`penalty=None`). And here is the results:

Table 3: Results of Logistic Regression using 5-fold cross validation

params	**mean fit time**	**mean score time**	**mean test score**	**mean train score**
{‘penalty’: ‘l2’, ‘C’: 10}	15.6687	0.0051	0.8700	0.8720
{‘penalty’: ‘l2’, ‘C’: 100}	31.1332	0.0053	0.8700	0.8725
{‘penalty’: ‘l2’, ‘C’: 1000}	35.6433	0.0054	0.8699	0.8725
{‘penalty’: None}	37.5202	0.0043	0.8697	0.8725
{‘penalty’: ‘l2’, ‘C’: 1}	7.0759	0.0053	0.8656	0.8677
{‘penalty’: ‘l2’, ‘C’: 0.1}	4.4298	0.0052	0.8350	0.8369

As C becomes larger, the regularization term (which penalizes large coefficients) has less influence. This allows the model to fit the training data more closely, potentially increasing the complexity of the model. Therefore, it takes longer time for the model with larger C to train and converge.

Decision Tree In the setup of Decision Tree, we tune the following parameters:

- `max_depth` : `[5, 10, 15, 20]`
- `min_samples_leaf` : `[5, 10, 20, 30, 50]`
- `criterion` : `[‘gini’, ‘entropy’]`

and here is the results:

Table 4: Results of Decision Tree using 5-fold cross validation

params	**mean fit time**	**mean score time**	**mean test score**	**mean train score**
{’min_samples_leaf’: 10, ’max_depth’: 10, ’criterion’: ’entropy’}	0.1437	0.0051	0.8831	0.9071
{’min_samples_leaf’: 20, ’max_depth’: 10, ’criterion’: ’entropy’}	0.1486	0.0055	0.8823	0.8976
{’min_samples_leaf’: 5, ’max_depth’: 10, ’criterion’: ’entropy’}	0.1459	0.0051	0.8820	0.9144

Table 4: Results of Decision Tree using 5-fold cross validation (continued)

params	mean fit time	mean score time	mean test score	mean train score
{'min_samples_leaf': 30, 'max_depth': 15, 'criterion': 'entropy'}	0.1273	0.0045	0.8819	0.8909
{'min_samples_leaf': 20, 'max_depth': 20, 'criterion': 'entropy'}	0.1650	0.0082	0.8817	0.8986
{'min_samples_leaf': 20, 'max_depth': 20, 'criterion': 'gini'}	0.1216	0.0053	0.8816	0.8974
{'min_samples_leaf': 20, 'max_depth': 15, 'criterion': 'gini'}	0.1245	0.0062	0.8816	0.8974
{'min_samples_leaf': 30, 'max_depth': 15, 'criterion': 'gini'}	0.1134	0.0053	0.8814	0.8912
{'min_samples_leaf': 30, 'max_depth': 10, 'criterion': 'gini'}	0.1076	0.0052	0.8811	0.8909
{'min_samples_leaf': 10, 'max_depth': 10, 'criterion': 'gini'}	0.1155	0.0049	0.8802	0.9060
{'min_samples_leaf': 10, 'max_depth': 20, 'criterion': 'gini'}	0.1457	0.0069	0.8787	0.9109
{'min_samples_leaf': 10, 'max_depth': 15, 'criterion': 'entropy'}	0.1530	0.0055	0.8781	0.9122
{'min_samples_leaf': 50, 'max_depth': 20, 'criterion': 'entropy'}	0.1302	0.0050	0.8766	0.8836

Table 4: Results of Decision Tree using 5-fold cross validation (continued)

params	mean fit time	mean score time	mean test score	mean train score
{'min_samples_leaf': 50, 'max_depth': 15, 'criterion': 'entropy'}	0.1273	0.0050	0.8766	0.8836
{'min_samples_leaf': 5, 'max_depth': 20, 'criterion': 'gini'}	0.2064	0.0095	0.8731	0.9283
{'min_samples_leaf': 50, 'max_depth': 10, 'criterion': 'gini'}	0.1010	0.0049	0.8728	0.8800
{'min_samples_leaf': 50, 'max_depth': 20, 'criterion': 'gini'}	0.1588	0.0074	0.8728	0.8800
{'min_samples_leaf': 5, 'max_depth': 5, 'criterion': 'gini'}	0.0691	0.0051	0.8435	0.8460
{'min_samples_leaf': 50, 'max_depth': 5, 'criterion': 'gini'}	0.0674	0.0049	0.8417	0.8436
{'min_samples_leaf': 50, 'max_depth': 5, 'criterion': 'entropy'}	0.0864	0.0050	0.8307	0.8345

In decision tree experiments, it was observed that setting a very low max_depth or a very high min_samples_leaf leads to poor training results.

Have not changed Random Forest and XGBoost

Random Forest In the setup of Random Forest, we tune the following parameters:

- `n_estimators` (T): `[100, 300, 500, 1000, 2000]`
- `max_depth` (maximum depth of a tree): `[10, 20, 30]`
- `min_samples_leaf` (minimum number of samples in a leaf): `[10, 20, 30]`
- `criterion`: `['gini', 'entropy']`

Table 5: Results of Random Forest using 5-fold cross validation

params	mean fit time	mean score time	mean test score	mean train score
{'n_estimators': 500, 'min_samples_leaf': 10, 'max_depth': 20, 'criterion': 'entropy'}	14.4901	0.4004	0.8965	0.9222
{'n_estimators': 1000, 'min_samples_leaf': 10, 'max_depth': 20, 'criterion': 'gini'}	24.5074	0.8482	0.8956	0.9216
{'n_estimators': 2000, 'min_samples_leaf': 10, 'max_depth': 10, 'criterion': 'entropy'}	56.5501	1.4682	0.8917	0.9141
{'n_estimators': 2000, 'min_samples_leaf': 10, 'max_depth': 10, 'criterion': 'gini'}	44.7706	1.5665	0.8904	0.9117
{'n_estimators': 2000, 'min_samples_leaf': 20, 'max_depth': 30, 'criterion': 'gini'}	44.7748	1.5418	0.8882	0.9064
{'n_estimators': 300, 'min_samples_leaf': 20, 'max_depth': 30, 'criterion': 'entropy'}	8.6514	0.2336	0.8880	0.9066
{'n_estimators': 500, 'min_samples_leaf': 20, 'max_depth': 20, 'criterion': 'gini'}	11.0895	0.3889	0.8879	0.9063

Table 5: Results of Random Forest using 5-fold cross validation (continued)

params	mean fit time	mean score time	mean test score	mean train score
{‘n_estimators’: 100, ‘min_samples_leaf’: 20, ‘max_depth’: 30, ‘criterion’: ’gini’}	2.2172	0.0799	0.8865	0.9044
{‘n_estimators’: 100, ‘min_samples_leaf’: 20, ‘max_depth’: 20, ‘criterion’: ’entropy’}	3.0099	0.0897	0.8863	0.9050
{‘n_estimators’: 100, ‘min_samples_leaf’: 10, ‘max_depth’: 10, ‘criterion’: ’gini’}	2.4740	0.0815	0.8858	0.9087
{‘n_estimators’: 2000, ‘min_samples_leaf’: 30, ‘max_depth’: 30, ‘criterion’: ’gini’}	42.4656	1.4651	0.8819	0.8963
{‘n_estimators’: 1000, ‘min_samples_leaf’: 30, ‘max_depth’: 20, ‘criterion’: ’gini’}	21.2953	0.7498	0.8818	0.8971
{‘n_estimators’: 300, ‘min_samples_leaf’: 30, ‘max_depth’: 20, ‘criterion’: ’gini’}	6.2479	0.2257	0.8811	0.8959
{‘n_estimators’: 100, ‘min_samples_leaf’: 30, ‘max_depth’: 30, ‘criterion’: ’gini’}	2.1063	0.0780	0.8807	0.8949

Table 5: Results of Random Forest using 5-fold cross validation (continued)

params	mean fit time	mean score time	mean test score	mean train score
{‘n_estimators’: 300, ‘min_samples_leaf’: 30, ‘max_depth’: 30, ‘criterion’: ’entropy’}	7.8691	0.2295	0.8805	0.8949
{‘n_estimators’: 500, ‘min_samples_leaf’: 30, ‘max_depth’: 10, ‘criterion’: ’gini’}	10.4965	0.3669	0.8795	0.8924
{‘n_estimators’: 300, ‘min_samples_leaf’: 30, ‘max_depth’: 10, ‘criterion’: ’entropy’}	8.0571	0.2142	0.8794	0.8935
{‘n_estimators’: 100, ‘min_samples_leaf’: 30, ‘max_depth’: 10, ‘criterion’: ’gini’}	2.0512	0.0782	0.8783	0.8904
{‘n_estimators’: 100, ‘min_samples_leaf’: 30, ‘max_depth’: 20, ‘criterion’: ’entropy’}	2.6312	0.0965	0.8777	0.8933
{‘n_estimators’: 100, ‘min_samples_leaf’: 30, ‘max_depth’: 10, ‘criterion’: ’entropy’}	2.6146	0.0756	0.8769	0.8914

In random forest experiments, model performance increased slightly when the number of estimators (n_estimators) was increased from 100 to 300. However, no clear performance difference was observed when varying n_estimators between 300 and 2000.

XGBoost To set up the XGBoost algorithm, we tune the following hyperparameters:

- `n_estimators` (number of trees): `[100, 200, 300, 500]`
- `max_depth` (of a tree) : `[10, 15, 20]`

Table 6: Results of XGBoost using 5-fold cross validation

params	**mean fit time**	**mean score time**	**mean test score**	**mean train score**
{‘n_estimators’: 500, ‘max_depth’: 15}	24.1672	0.7437	0.9057	1.0000
{‘n_estimators’: 300, ‘max_depth’: 15}	16.4752	0.5553	0.9055	1.0000
{‘n_estimators’: 100, ‘max_depth’: 10}	6.3175	0.1774	0.9051	0.9999
{‘n_estimators’: 200, ‘max_depth’: 15}	12.2354	0.3745	0.9051	1.0000
{‘n_estimators’: 100, ‘max_depth’: 15}	7.1653	0.2139	0.9048	1.0000
{‘n_estimators’: 500, ‘max_depth’: 20}	22.8552	0.6701	0.9047	1.0000
{‘n_estimators’: 300, ‘max_depth’: 20}	16.3073	0.5154	0.9046	1.0000
{‘n_estimators’: 300, ‘max_depth’: 10}	15.9453	0.4860	0.9043	1.0000
{‘n_estimators’: 200, ‘max_depth’: 20}	12.4350	0.3785	0.9041	1.0000
{‘n_estimators’: 100, ‘max_depth’: 20}	7.5500	0.2092	0.9037	1.0000
{‘n_estimators’: 200, ‘max_depth’: 10}	11.2365	0.3470	0.9035	1.0000
{‘n_estimators’: 500, ‘max_depth’: 10}	24.2957	0.7151	0.9029	1.0000

In XGBoost experiments, the training score is perfect, meaning the model fits the data completely. Additionally, there is a little difference in the test scores among different settings.

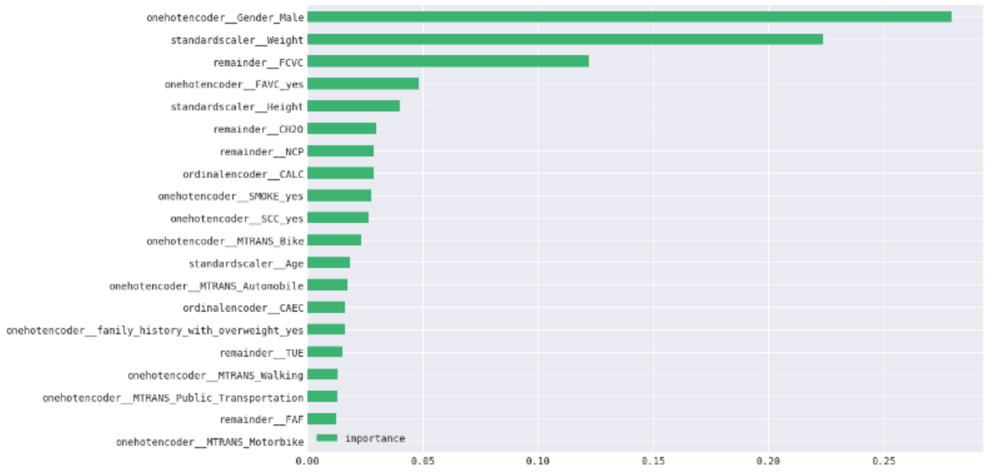


Figure 14: Feature importance of XGBoost

In Figure 14, some importance features are ‘Gender’, ‘Weight’ and ‘FCVC’. In contrast, ‘MTRANS’, ‘FAF’ are less useful to our model.

Voting Classifier: We choose `XGBClassifier(n_estimators = 500, max_depth = 15)` - the best model in evaluation process, `RandomForestClassifier(n_estimators = 500, min_samples_leaf = 10, max_depth = 20, criterion = ‘entropy’)` - the best model that is not XGBoost, and `XGBClassifier(n_estimators = 100, max_depth = 10)` - not the second best model, but the max depth are different.

5.4 Results and Explanation

In this section, we will present the results of each model for training - validation set and make comparisons. After that, we pick the best model and evaluate it on test set. After applying various techniques and experimenting with different parameter settings, here is the summary of the results of different models with the best-performing parameter settings we have tried.

Table 7: Model performance metrics on training - validation set

params	**mean fit time**	**mean score time**	**mean test score**	**mean train score**	**model name**
{‘p’: 1, ‘n_neighbors’: 20}	0.0117	3.4113	0.8297	0.8484	knn
{‘penalty’: ‘l2’, ‘C’: 10}	15.6687	0.0051	0.8700	0.8720	logistic_regression
{‘penalty’: None}	75.1169	0.0103	0.8754	0.8782	logistic_regression feateng
{‘min_samples_leaf’: 10, ‘max_depth’: 10, ‘criterion’: ‘entropy’}	0.1437	0.0051	0.8831	0.9071	decision_tree

Table 7: Model performance metrics training - validation set (continued)

params	mean fit time	mean score time	mean test score	mean train score	model name
{‘n_estimators’: 500, ‘min_samples_leaf’: 10, ‘max_depth’: 20, ‘criterion’: ‘entropy’}	14.4901	0.4004	0.8965	0.9222	random_forest
{‘n_estimators’: 500, ‘max_depth’: 15}	24.1672	0.7437	0.9057	1.0000	xg_boost
XGBClassifier (n_estimators = 500, max_depth = 15); RandomForestClassifier(n_estimators = 500, min_samples_leaf = 10, max_depth = 20, criterion = entropy); XGBClassifier(n_estimators = 100, max_depth = 10)	43.9071	1.2480	0.9072	1.0000	voting_classifier

The above table presents performance of our 5 models, sorting by ‘mean test score’ in ascending order. First of all, KNN, the most simple model, achieves a decent test score with 0.8297 but the drawback is its longest prediction time (3.4113 seconds). The fastest model (0.0051 seconds) with high result (0.8831) is decision tree with min_samples_leaf=10 and max_depth=10. Additionally, using many estimators to predict at the same time, ensemble models, namely random forest, XGBoost, voting classifier, gain an improvement in their performance. Despite their slightly long training time, the prediction time is still acceptable, around 1 second, therefore can be applied in the real life. We also apply feature engineering for the logistic regression. However, the model yields a small gain , while training time is 5 times longer.

After conducting experiments and comparison, we decide to choose the model with the best cross validation result, which is the voting classifier. Finally, this model will be evaluated on the test set, providing us an accurate assessment when working with unseen future data.

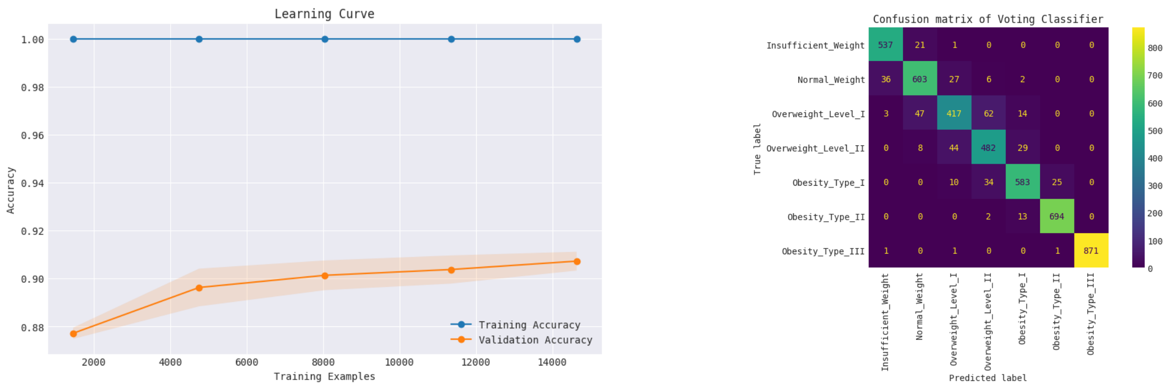


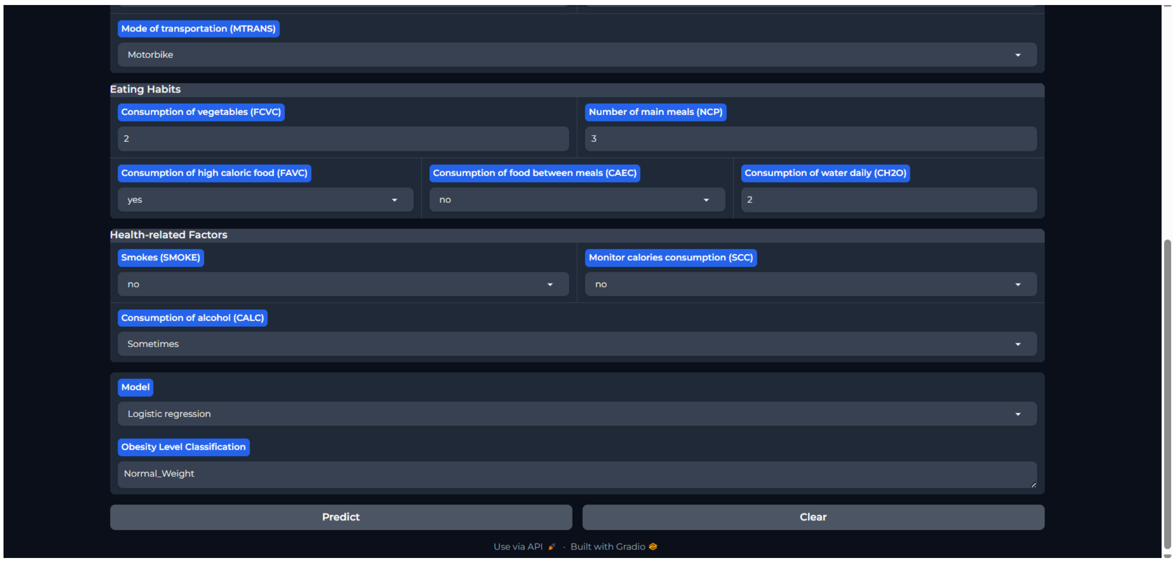
Figure 15: Learning curve and confusion matrix for Voting classifier when running on **Test set**

In Figure 15, at the learning curve, we can see that the training accuracy remains consistently at 1.0 across all training set sizes, while validation accuracy starts around 0.88 with a smaller training set and slightly increases as the number of training examples increases, reaching just above 0.90 with the largest training set. This gap of 2 lines infers the possible overfitting problem. To mitigate this issue, one of our future work is to collect more data and apply regularization techniques. However, at this time, the validation accuracy is relatively high, hence this result is acceptable.

Looking at the confusion matrix, voting classifier model demonstrates strong performance across most classes, with particularly high accuracy for extreme categories like Insufficient Weight and Obesity Type III. However, there is remarkable confusion between adjacent weight categories, especially within the overweight and lower obesity levels. This suggests that while the classifier is generally effective, there is room for improvement in distinguishing between categories that are closer in the obesity spectrum.

5.5 GUI implementation

We implemented the web deployment of our project using Gradio to demonstrate the performance of the models in our project by entering all the given input in the website following the instruction, after that, it will display a predicted label of the obesity level. To begin using our user interface tool, please review and follow the provided instructions in “Readme.md” file carefully. An illustrative example of our application can be seen in the below image:



Mode of transportation (MTRANS): Motorbike

Eating Habits

Consumption of vegetables (FCVC): 2

Number of main meals (NCP): 3

Consumption of high caloric food (FAVC): yes

Consumption of food between meals (CAEC): no

Consumption of water daily (CH2O): 2

Health-related Factors

Smokes (SMOKE): no

Monitor calories consumption (SCC): no

Consumption of alcohol (CALC): Sometimes

Model: Logistic regression

Obesity Level Classification: Normal_Weight

Figure 16: Prediction of an instance and logistic regression model in web application

6 Conclusion and Possible Extension

So far, we have introduced and solved our prediction problem by different techniques, such as logistic regression, random forest or XGBoost. The model that shows the best classification performance, in general, is the voting classfier combining random forest and xgboost which performs the most effectively with both training and testing set on the dataset.

One of the most challenging aspects of solving our problem was determining the appropriate training method for our dataset, as well as selecting the models and techniques for workload assignments among team members. Ultimately, we opted for the aforementioned techniques and models discussed in earlier sections. And finally, we decided to follow an aforementioned process.

As we can see from the table of results of different models, ensemble model, especially voting classfier yield much better results in term of errors for this type of problem. Therefore, given more time on this project, we would have found better training strategy for each models and further explored new AI models. Additionally, we would have delved into the application of self-supervised and semi-supervised learning methods on our preprocessed dataset.

All the codes, notebooks, trained weights model files for the project with detailed instructions can be found in the submited zip file.

7 Member Assignments

- Nguyen Chi Long - 20210553: Model, theory.
- Ngo Xuan Bach - 20215181: Data preprocessing, model.
- Le Xuan Hieu - 20215201: Model, evaluation.
- Dinh Viet Quang - 20215235: EDA, model deployment.
- Nguyen Viet Thuan - 20210826: EDA, evaluation.

References

[1] Hust.edu.vn, 2023.

[2] AravindPCoder. Obesity or cvd risk (classify/regressor/cluster). `https://www.kaggle.com/datasets/aravindpcoder/obesity-or-cvd-risk-classifyregressorcluster`, 2023.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[4] Kaggle. Multi-class prediction of obesity risk. `https://www.kaggle.com/competitions/playground-series-s4e2`, 2024.

[5] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, 2017.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.