1. **Data information**The dataset contains 18 columns in total.

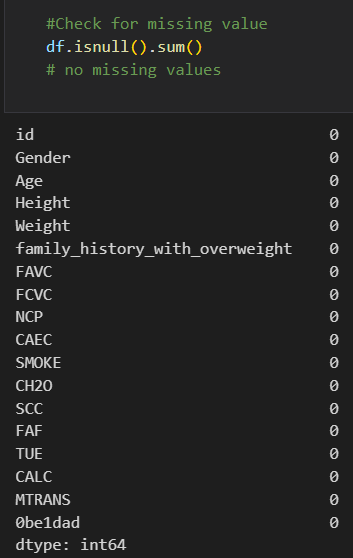
* Numerical attributes: id, Age, Height, Weight, family\_history\_with\_overweight, FAVC (Frequent consumption of high-caloric food), FCVC (Frequency of consumption of vegetables), NCP (Number of main meals), CH2O (Daily water consumption), FAF (Physical activity frequency), TUE (Time spent using technological devices), SMOKE, SCC (Caloric beverages consumption)
* Categorical attributes: Gender, CAEC (Consumption of food between meals), CALC (Consumption of alcohol), MTRANS (Mode of transportation), 0be1dad (Target variable representing obesity level)
* **The goal of predicting mental health (binary) based on people’s occupation**

**+ Important features:** Gender, Age, Height, Weight, family\_history\_with\_overweight, FAVC, FCVC, NCP, CAEC, SMOKE, CH2O, SCC, FAF, TUE, CALC, MTRANS

* **Age**: Younger individuals often have higher metabolism and more active lifestyles compared to older adults. Obesity prevalence tends to increase with age as metabolism slows. Older individuals might adopt sedentary lifestyles, contributing to weight gain. Childhood obesity can predispose individuals to adult obesity due to habits formed early in life.
* **Height**: is essential to calculate Body Mass Index alongside weight. Taller individuals may naturally weigh more without being obese. Height helps standardize weight assessments to account for body proportions.
* **Weight**: a direct indicator of an individual’s physical mass. Excessive weight relative to height indicates overweight or obesity. Weight trends (gains or losses) reflect changes in dietary and lifestyle habits.
* **family\_history\_with\_overweight**: reflects genetic predispositions and shared environmental factors. Individuals from families with overweight members are more likely to experience similar issues due to genetic factors like slower metabolism. Shared habits within families also play a role.
* **FAVC**: High-calorie foods contribute directly to excessive energy intake. Regular consumption of calorie-dense foods like fast food or desserts leads to energy imbalances. Poor portion control exacerbates weight gain over time.
* **FCVC**: Vegetables are low in calories and high in fiber, promoting fullness and better digestion. Low vegetable consumption often correlates with poor dietary habits. High fiber intake helps in weight management by reducing hunger and calorie intake.
* **NCP**: Eating frequency impacts energy intake and metabolic rate. Skipping meals may lead to overeating later in the day, increasing overall calorie consumption. More structured meal patterns can stabilize metabolism and prevent binge eating.
* **CH2O**: Water is critical for metabolic processes and maintaining hydration. Low water intake can lead to overeating, as thirst is often mistaken for hunger. Adequate water consumption supports digestion and calorie burning.
* **FAF**: Regular exercise helps burn calories and improves overall health. Low physical activity levels are a leading cause of energy imbalance. Active individuals maintain healthier weights due to increased energy expenditure.
* **TUE**: This reflects sedentary behavior, like prolonged screen time or gaming. High TUE is associated with physical inactivity and lower energy expenditure. Screen time can promote unhealthy snacking and irregular eating patterns.
* **SMOKE**: affects appetite, metabolism, and lifestyle. Smokers often have lower body weight, but quitting smoking can lead to weight gain. Smoking-related stress or habits might indirectly affect dietary choices and activity levels.
* **SCC**: Sugary drinks add hidden calories to the diet. Caloric beverages like sodas, energy drinks, and sweetened coffee contribute to weight gain without providing satiety. Habitual consumption leads to significant calorie accumulation over time.
* **Gender**: influences hormonal balance, fat distribution, and metabolism. Men tend to store fat in the abdomen, while women are more likely to gain weight in the hips and thighs.
* **CAEC**: Snacking habits impact overall calorie intake and meal structure. Unhealthy snacks (e.g., chips, candy) add extra calories and disrupt energy balance. Frequent snacking can be a sign of emotional eating or poor meal planning.
* **CALC**: Alcohol is calorie-dense and impacts metabolism and decision-making. Regular alcohol consumption adds “empty calories” to the diet. Drinking often leads to overeating or choosing unhealthy food options.
* **MTRANS**: Reflects daily physical activity levels. Active modes of transportation (e.g., walking, cycling) help burn calories and improve fitness. Sedentary transportation modes (e.g., cars, public transport) reduce energy expenditure.

1. **Data preprocessing**
2. **Handling Missing Values**

* Check for null data: No missing data in dataset



1. **Remove duplicated Rows**

No duplicated rows to remove

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động**Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, Đồ họa

Mô tả được tạo tự động

1. **Outlier Detection and Removal**

We check outliers of numerical attributes by applying IQR:

import numpy as np

numerical\_columns = df.select\_dtypes(include=[np.number]).columns

# Number of numerical features

num\_features = len(numerical\_columns)

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Function to detect outliers based on IQR

def detect\_outliers(df, numerical\_columns):

    outliers = {}

    for col in numerical\_columns:

        Q1 = df[col].quantile(0.25)

        Q3 = df[col].quantile(0.75)

        IQR = Q3 - Q1

        lower\_bound = Q1 - 1.5 \* IQR

        upper\_bound = Q3 + 1.5 \* IQR

        outliers[col] = df[(df[col] < lower\_bound) | (df[col] > upper\_bound)].index

    return outliers

# Function to remove outliers

def remove\_outliers(df, numerical\_columns):

    for col in numerical\_columns:

        Q1 = df[col].quantile(0.25)

        Q3 = df[col].quantile(0.75)

        IQR = Q3 - Q1

        lower\_bound = Q1 - 1.5 \* IQR

        upper\_bound = Q3 + 1.5 \* IQR

        df = df[(df[col] >= lower\_bound) & (df[col] <= upper\_bound)]

    return df

# Detect outliers in numerical columns

outliers = detect\_outliers(df, numerical\_columns)

# Display outliers

for col, indices in outliers.items():

    print(f"\nOutliers in {col}:")

    print(df.loc[indices])

# Visualize outliers with boxplots

rows = len(numerical\_columns) // 4 + 1

cols = 4

plt.figure(figsize=(16, rows \* 4))

for i, col in enumerate(numerical\_columns, 1):

    plt.subplot(rows, cols, i)

    sns.boxplot(data=df, y=col)

    plt.tight\_layout()

plt.show()

# Remove outliers from the dataset

cleaned\_df = remove\_outliers(df, numerical\_columns)

# Display the cleaned dataset

print("\nDataset after removing outliers:")

print(cleaned\_df)

As we can see data’s outliers. Box plot Ảnh có chứa Hình chữ nhật, hình vuông, ảnh chụp màn hình, Nhiều màu sắc

Mô tả được tạo tự động

1. **Encoding Categorical Variables**

**CAEC** and **CALC** is ordinal attributes, then we use label encoding

**MTRANS** and **Gender** is nominal attributes, then we use one-hot encoding

# Label Encoding for ordinal attributes

CAEC\_order = {'0': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3}

CALC\_order = {'0': 0, 'Sometimes': 1, 'Frequently': 2}

cleaned\_df['CAEC\_Encoded'] = cleaned\_df['CAEC'].map(CAEC\_order)

cleaned\_df['CALC\_Encoded'] = cleaned\_df['CALC'].map(CAEC\_order)

# One-Hot Encoding for nominal attributes

cleaned\_df = pd.get\_dummies(cleaned\_df, columns=['MTRANS', 'Gender'], drop\_first=False)

To built model, we convert all True/False values into to numeric

columns = ['MTRANS\_Automobile','MTRANS\_Bike', 'MTRANS\_Motorbike', 'MTRANS\_Public\_Transportation', 'MTRANS\_Walking','Gender\_Female', 'Gender\_Male']

for c in columns:

    cleaned\_df[c] = cleaned\_df[c].apply(lambda x: 1 if x else 0)

1. **Normallization/Scaling**

If the dataset has features with different scales like "Age" (ranging from 20 to 60), the model may give more importance to features with larger numerical values. Normalization/scaling transforms the features so that they are on a similar scale, making sure no single feature dominates the others due to its larger scale.

We are scaling the numerical features Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, sách

Mô tả được tạo tự động

Plot before scaling: Ảnh có chứa tòa nhà chọc trời, đường chân trời, bầu trời, Tòa tháp

Mô tả được tạo tự động

Plot after scaling: Ảnh có chứa tòa nhà chọc trời, bầu trời, đường chân trời, tòa nhà

Mô tả được tạo tự động

1. **Correlation Matrix**

**Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, hình vuông

Mô tả được tạo tự động**

1. **Drop, Move and Rename column**

Dropping

+ original categorical columns after encoding: CAEC, CALC

+ unuseful columns: id

+ constant colum: family\_history\_with\_overweight, FAVC, NCP, SMOKE, SCC

A constant column will have the same value for every observation. This means the model cannot use it to distinguish between different instances in the dataset. Moreover, constant columns are effectively redundant. Their presence might lead to overfitting or unnecessarily increase the dimensionality of the model. Constant columns generally improves accuracy by making your model more efficient, reducing overfitting, and focusing on more relevant features

Move and Rename column: Move the '0be1dad' (target) column to the last position and rename the column

# Move the '0be1dad' column to the last position

target\_column = '0be1dad'

cleaned\_df = cleaned\_df[[col for col in cleaned\_df if col != target\_column] + [target\_column]]

# Rename the column

cleaned\_df = cleaned\_df.rename(columns={target\_column: 'class'})

Rename values of class for shorter name:

mapping = {

    'Overweight\_Level\_II': 'OvlII',

    '0rmal\_Weight': 'Normal',

    'Insufficient\_Weight': 'Insuff',

    'Obesity\_Type\_III': 'ObIII',

    'Obesity\_Type\_II': 'ObII',

    'Overweight\_Level\_I': 'OvlI',

    'Obesity\_Type\_I': 'ObI'

}

cleaned\_df['class'] = cleaned\_df['class'].map(mapping)

1. **Convert to ARFF**

* Spliting data:
* **Validation\_data:** 20% of the original data
* **train\_data:** 80% of the original data

from sklearn.model\_selection import train\_test\_split

train\_data, validation\_data = train\_test\_split(cleaned\_df, test\_size=0.2, random\_state=42)

* Convert to ARFF file

from scipy.io import arff

import pandas as pd

# Function to convert DataFrame to ARFF format and save

def save\_to\_arff(cleaned\_df, filename, relation\_name="data"):

    with open(filename, 'w') as f:

        # Write the ARFF header

        f.write(f"@relation {relation\_name}\n\n")

        # Define attributes

        for col in cleaned\_df.columns:

            f.write(f"@attribute {col} real\n")

        # Write the data section

        f.write("\n@data\n")

        for row in cleaned\_df.values:

            f.write(",".join(map(str, row)) + "\n")

save\_to\_arff(train\_data, 'train\_data.arff', "Obesity\_Risk\_train\_data")

save\_to\_arff(validation\_data, 'validation\_data.arff', "Obesity\_Risk\_validation\_data")

* Save cleaned data to csv file:

cleaned\_df.to\_csv('cleaned\_data.csv', index=False)