



Vietnamese-German University



VIETNAMESE – GERMAN UNIVERSITY  
DEPARTMENT OF COMPUTER SCIENCE

FRANKFURT UNIVERSITY OF APPLIED SCIENCES  
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

---

**LOGISTIC REGRESSION ANALYSIS TO PREDICT OBESITY LEVELS:  
INSIGHTS FROM HEALTH SURVEY DATA**

---

**Author:** Le Dang Nhat Duong (Leader)  
Nguyen Ngoc Hoang Nam  
Cao Thang Long  
Vuong Thieu Luan

**Supervisor:** Prof. Christina Andersson

**PROJECT REPORT**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF BACHELOR ENGINEERING IN STUDY PROGRAM COMPUTER  
SCIENCE, VIETNAMESE-GERMAN UNIVERSITY, 2024

BINH DUONG, VIETNAM

# Contents

<b>0</b>	<b>Abstract</b>	<b>6</b>
<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Purpose of the Study . . . . .	7
<b>2</b>	<b>Methodology</b>	<b>8</b>
2.1	Data Collection . . . . .	8
2.1.1	Survey Design . . . . .	8
2.1.2	Sampling Procedure . . . . .	9
2.1.3	Ethical Considerations . . . . .	10
2.2	Software . . . . .	11
2.3	Libraries . . . . .	11
<b>3</b>	<b>Data Preparation</b>	<b>12</b>
3.1	Data Cleaning . . . . .	12
3.1.1	What is Data Cleaning? . . . . .	12
3.1.2	Why is Data Cleaning Important? . . . . .	12
3.1.3	Steps to Perform Data Cleanliness . . . . .	12
3.1.4	Implementation for Database Cleaning . . . . .	13
3.2	Data Transformation . . . . .	14
3.2.1	Encoding Categorical Data . . . . .	14
<b>4</b>	<b>Logistic Regression</b>	<b>16</b>
4.1	What is Logistic Regression? . . . . .	16
4.2	Key Points . . . . .	16
4.3	Logistic Function – Sigmoid Function . . . . .	16
4.4	How does Logistic Regression work? . . . . .	17
4.4.1	Sigmoid Function . . . . .	17
4.4.2	Logistic Regression Equation . . . . .	18
4.4.3	Likelihood Function for Logistic Regression . . . . .	18
4.4.4	Gradient of the log-likelihood function . . . . .	19
<b>5</b>	<b>Evaluation method</b>	<b>20</b>
5.0.1	Confusion Matrix . . . . .	20
5.0.2	AUC-ROC curve . . . . .	21
<b>6</b>	<b>Results</b>	<b>22</b>
6.1	Forward Feature Selection . . . . .	22

6.2	Logistics Regression Model . . . . .	22
6.2.1	Splitting data . . . . .	22
6.2.2	Training data and Predictions . . . . .	23
6.3	Confusion Matrix . . . . .	23
6.4	Receiver Operating Characteristic (ROC) curve . . . . .	24
<b>7</b>	<b>Conclusion</b>	<b>26</b>
7.1	Key Takeaways . . . . .	26
7.2	Final Remarks . . . . .	27
7.3	Future Directions . . . . .	27
<b>8</b>	<b>References</b>	<b>28</b>
<b>9</b>	<b>Appendix</b>	<b>29</b>
9.1	Survey Questionnaire . . . . .	29
9.2	Detailed Statistical Tables . . . . .	32
9.3	Python code . . . . .	32
<b>10</b>	<b>Project Contributions Table</b>	<b>44</b>

# List of Figures

6.1	Figure Selection . . . . .	22
6.2	Confusion Matrix . . . . .	23
6.3	Accuracy of the Logistic Regression model . . . . .	24
6.4	Evaluation Metric . . . . .	24
6.5	Receiver Operating Characteristic Curve . . . . .	24
9.1	Age Question . . . . .	29
9.2	Gender Question . . . . .	29
9.3	Height Question . . . . .	30
9.4	Weigth Question . . . . .	30
9.5	Exercise frequency Question . . . . .	30
9.6	Average Sleep Hours Question . . . . .	31
9.7	Daily servings of fruits and vegetables Question . . . . .	31
9.8	Frequency of sugary drink consumption Question . . . . .	31
9.9	Frequency of fast consumption Question . . . . .	32

# List of Tables

10.1 Contribution of Team Members in the Project . . . . . 44

# Chapter 0

## Abstract

We present to you our predictive model that predicts obesity based on logistic regression, a supervised machine learning algorithm that performs binary classification tasks by predicting the probability of an outcome, event, or observation. This model is designed to analyze and predict obesity by utilizing a wide range of data collected through comprehensive surveys.

Imagine a healthcare professional trying to identify individuals at risk of obesity in a community based on their lifestyle habits and physical attributes. Instead of manually analyzing survey responses and demographic data, we use a predictive model powered by logistic regression. This model processes data from the survey to determine an individual's obesity status. By providing quick and accurate predictions, the model enables healthcare providers to prioritize early interventions and design personalized health plans to prevent obesity-related complications.

Our objective is to develop a detailed and reliable model capable of predicting an individual's likelihood of being classified as obese. To achieve this, we examine several important variables, including demographic factors like age and gender, physical measurements such as height, weight, and BMI, and lifestyle behaviors such as the regularity of physical exercise, dietary choices like fruit and vegetable consumption, and the frequency of consuming sugary drinks and fast food. By integrating and analyzing these variables, our model aims to provide accurate predictions that can be used to identify at-risk individuals and guide preventive health strategies.

# Chapter 1

## Introduction

### 1.1 Purpose of the Study

The purpose of this study is to develop a robust and reliable predictive model to assess the likelihood of obesity in individuals, utilizing logistic regression—a supervised machine learning algorithm. By integrating and analyzing a range of variables such as demographic data, physical measurements, and lifestyle behaviors, the model aims to accurately predict obesity status. This study will provide healthcare professionals with a tool for identifying at-risk individuals, facilitating early interventions, and guiding the design of personalized health plans. Ultimately, the goal is to contribute to effective obesity prevention strategies and improve public health outcomes.

# Chapter 2

## Methodology

### 2.1 Data Collection

#### 2.1.1 Survey Design

Online surveys are the most cost-effective method for data collection, reaching more individuals and handling multiple questions better than telephone or face-to-face surveys. They are preferred for their efficiency and ability to outperform traditional methods.

With advanced technologies like branching logic, online surveys offer greater accuracy, are easy to deploy, and save respondents' time. They are low-cost and provide real-time results, enabling quick analysis and decision-making.

#### How to Design a Survey

1. Define the Purpose of Your Survey

Clarify the survey's goal, identify insights you aim to gather, and predict potential outcomes. Use existing surveys for inspiration and focus on specific research needs.

2. Identify Your Target Population

Determine who will participate and ensure the sample size is sufficient to answer your research questions effectively.

3. Choose the Right Survey Design and Method

Decide on quantitative or qualitative approaches, and whether to collect data at one point (cross-sectional) or over time (longitudinal).

4. Develop Your Survey Questions

Create clear, purposeful questions. Use both closed-ended (for precise data) and open-ended (for detailed responses) formats.

5. Administer Your Survey

Share the survey through suitable channels (e.g., social media, email, or interviews) to ensure it reaches the right participants.

6. Analyze the Data Collected

Use statistical tools for quantitative data and thematic analysis for qualitative responses to extract meaningful insights.



## 7. Draw Conclusions and Take Action

Evaluate whether goals were met, apply findings to make decisions, and use the results as a basis for future research.

## Helpful Survey Tips

- Keep Surveys Short
- Align Questions with Your Survey Aim
- Focus on Clear and Concise Wording
- Balance Open-Ended and Closed-Ended Questions
- Test Your Survey Before Launch
- Monitor and Optimize During Data Collection

## 2.1.2 Sampling Procedure

The 5 most important steps in a sampling procedure for survey data collection aimed at creating a predictive model are:

### 1. Define the Target Population

- Why Important: Ensures the data collected is relevant to the problem the model is solving.
- Key Action: Clearly identify the characteristics and scope of the population your predictive model will apply to.

### 2. Choose the Sampling Method

- Why Important: Determines how well the sample represents the target population and impacts model accuracy.
- Key Action: Use probability sampling (e.g., random or stratified sampling) if generalization is needed, or non-probability sampling for specific groups.

### 3. Determine Sample Size

- Why Important: Ensures sufficient data for reliable predictions while avoiding unnecessary resource expenditure.
- Key Action: Calculate the sample size based on population size, desired confidence level, and margin of error.

### 4. Ensure Representativeness

- Why Important: Reduces bias and improves the model's applicability to the entire population.
- Key Action: Stratify the sample or monitor demographic proportions to match the target population.

## 5. Collect Relevant Features

- Why Important: The model's accuracy depends on the availability of key variables (features) influencing the outcome.
- Key Action: Align survey questions with the variables needed for your predictive model, ensuring completeness.

### 2.1.3 Ethical Considerations

#### 1. Informed Consent

Participants must understand the purpose of the survey, how their data will be used, and agree to participate voluntarily.

#### 2. Privacy and Confidentiality

Protect participants' personal information and ensure their data is not shared without permission.

#### 3. Avoiding Harm or Discomfort

The survey should not cause emotional, psychological, or social distress.

#### 4. Transparency and Honesty

Be truthful about the survey's purpose and avoid deception.

#### 5. Voluntary Participation

Participants should feel free to join or withdraw without facing negative consequences.

#### 6. Inclusivity and Accessibility

Ensure the survey is accessible to all intended participants.

#### 7. Bias Minimization

Design the survey to avoid unintentional bias that might skew results.

#### 8. Compliance with Legal and Ethical Standards

Follow regulations such as GDPR (in Europe), HIPAA (for health data), or local data protection laws.

#### 9. Debriefing Participants

Offer participants a summary of the survey's findings and their role in the project if possible.

## 10. Data Usage and Retention

Use data only for the stated purpose and dispose of it responsibly once it is no longer needed.

## 2.2 Software

- **Google Form:** We use it to create a survey and collect data from students in VGU University.
- **Google Colab:** We use it to do all the analytics of the project (data preprocessing, data mining, data analysis)

## 2.3 Libraries

- **Numpy:** A fundamental library for numerical computing in Python, providing support for arrays, matrices, and mathematical operations on these structures efficiently.
- **Seaborn:** A data visualization library built on top of Matplotlib, designed for creating attractive and informative statistical graphics with ease.
- **Pandas:** A library providing high-performance data structures like DataFrames and tools for data manipulation, analysis, and cleaning in Python.
- **Matplotlib.pyplot:** A module within Matplotlib that provides an interface for creating static, animated, and interactive visualizations in Python.
- **Sklearn.preprocessing:** A module for preprocessing data in machine learning pipelines, including scaling, normalization, and encoding features.
- **Scipy.stats:** A module in SciPy for statistical functions, probability distributions, and hypothesis testing.
- **Sklearn.linear-model:** A module in Scikit-learn that provides linear models for regression and classification tasks, such as Linear Regression, Logistic Regression, and Ridge Regression.
- **Sklearn.model-selection:** A module for splitting data into training and testing sets, performing cross-validation, and selecting the best model.
- **Sklearn.metrics:** A module for evaluating the performance of machine learning models through metrics like accuracy, precision, recall, and F1 score.

# Chapter 3

## Data Preparation

### 3.1 Data Cleaning

#### 3.1.1 What is Data Cleaning?

Data cleaning is a key step in machine learning that removes missing, duplicate, or unnecessary data to ensure accuracy, consistency, and error-free information. It is essential for model performance, as poor data can negatively affect results. Data scientists focus on this process, as high-quality data is often more important than complex algorithms. Data cleaning, also known as data cleansing or preparation, it addresses errors and inconsistencies to improve data quality and reliability.

#### 3.1.2 Why is Data Cleaning Important?

Data cleaning ensures a dataset's accuracy and reliability, which is crucial for making informed decisions. Inaccuracies and inconsistencies can distort results, while clean data enhances modeling, pattern detection, and the interpretability of findings, leading to better insights.

#### 3.1.3 Steps to Perform Data Cleanliness

Data cleaning is a systematic method for identifying and correcting mistakes, inconsistencies, and inaccuracies in a dataset. The following are necessary processes for data cleaning:

- **Remove Unwanted Observations:** Eliminate duplicates or irrelevant data to improve dataset quality.
- **Fixing Structure Errors:** Standardize formats and resolve inconsistencies for more accurate analysis.
- **Managing Unwanted outliers:** Address extreme data points, either by removing or adjusting them.
- **Handling Missing Data:** Use methods like imputation or removal to deal with missing values, ensuring a comprehensive dataset.

### 3.1.4 Implementation for Database Cleaning

Let's understand each step for Database Cleaning. Below are the necessary steps:

1. **Import the necessary libraries(.e.g pandas, numpy...)**
2. **Load the dataset**
3. **Check the data information using df.info()**
4. **Data Inspection and Exploration**
5. **Check the Categorical and Numerical Columns**
6. **Removal of all Above Unwanted Observations**

This includes removing duplicate, redundant, and unnecessary values from your dataset. Duplicate observations are most common during data gathering, and irrelevant observations are those that do not relate to the specific problem that you are attempting to address.

- Redundant observations have a significant impact on efficiency because the data repeats and may contribute to the correct or incorrect side, resulting in inaccurate results.
- Irrelevant observations are any type of data that serves no purpose to us and can be eliminated immediately.

We must now decide which factor is relevant to our discussion based on the subject of analysis.

#### 7. Handling Missing Data

Missing data is common in real-world datasets due to factors like human errors or system failures. It can be handled through techniques such as imputation, deletion, or substitution.

To check the percentage of missing values, use `df.isnull()` to detect null values and `.sum()` to calculate the percentage. Missing data should be addressed carefully, as it may indicate important information, rather than just being ignored or removed.

The two most common ways to deal with missing data are:

- Dropping Observations with missing values.
  - The fact that the value was missing may be informative in itself.
  - Plus, in the real world, you often need to make predictions on new data even if some of the features are missing!
- Imputing the missing values from past observations.

- Again, “missingness” is almost always informative in itself, and you should tell your algorithm if a value was missing.
- Even if you build a model to impute your values, you’re not adding any real information. You’re just reinforcing the patterns already provided by other features.

We can use Mean imputation or Median imputations for the case.

## 8. Handling Outlier

Outliers are extreme values that differ significantly from the rest of the data and can negatively impact model performance. They can be handled using transformation, interpolation, or clustering. A box plot is often used to detect outliers, showing the distribution, median, quartiles, and the interquartile range (IQR). Outliers are points outside the whiskers, which extend 1.5 times the IQR. Box plots help identify outliers and skewness in the data.

## 3.2 Data Transformation

To make data better suited for analysis, data transformation entails changing the data’s format. The data can be transformed using methods like encoding, scaling, or normalization.

### 3.2.1 Encoding Categorical Data

When pursuing a machine learning task, we are likely to encounter categorical data. Categorical variables, unlike numerical data, are neither understood or worked with directly by computers. As a result, our objective is to develop a mechanism to "encode" category data, transforming it into another format that can be used in our model.

#### What is Categorical Data?

Categorical data, unlike numerical data, only takes a finite number of fixed values. For example, in a user dataset, gender is a categorical feature because it only allows one of two values: male or female. Supermarket product data may have a categorical element that supports several values, such as drinks, sweets, veggies, and personal items. Categorical data is frequently provided in text form.

Categorical data is often divided into 2 types: nominal and ordinal data:

- Nominal data: data that is labeled completely without any order or hierarchy. For example, Male/Female - sunny/rainy/cloudy weather - names of countries, cities, etc.
- Ordinal data: opposite to nominal data, is data that is arranged/classified in a certain order. For example, good/average/bad economic situation - clothing sizes XS/ S/ M/ L/ XL, etc.

## Ways to work with categorical data

In this project we mainly use `OrdinalEncoder` and `OneHotEncoder`

- Integer encoding/ Ordinal encoding: Converting category values to numbers is the most natural approach, mapping each category to an integer. For example, drinking products are ranked zero, spices are one, food is two, veggies are three, and so on. However, this secretly assigns these categories a value, a rank that is not necessarily true in reality (nominal features), which can lead to inaccuracies when calculating in the model.
- One-hot encoding is one of the most common methods for encoding categorical information. As a result, each category value will correspond to a one-hot vector of  $k$  elements ( $k$  is the number of distinct values). The process is as follows: for each different value of the categorical feature, we generate a new feature, such as `Category_drink`, `Category_food`, and so on. Each new category will be allocated one of two values: 0 or 1. If the product belongs to a category, its value will be one. These newly formed features are known as Dummy variables. One-hot encoding has the disadvantage of being memory inefficient, prone to memory overflow, slowing down and reducing learning efficiency.

# Chapter 4

## Logistic Regression

### 4.1 What is Logistic Regression?

Logistic regression is used for binary classification, and we use the sigmoid function, which accepts input as independent variables and returns a probability value between 0 and 1. For example, we have two classes: Class 0 and Class 1. If the value of the logistic function for an input is larger than 0.5 (threshold value), it is classified as Class 1, otherwise as Class 0. It is known as regression since it is an extension of linear regression, however it is mostly used for classification tasks.

### 4.2 Key Points

- Logistic regression predicts the value of a categorical dependent variable. As a result, the value must be categorical or discrete.
- It can be Yes or No, 0 or 1, true or false, and so on, but instead of providing exact values such as 0 and 1, it provides probability values between 0 and 1.
- In logistic regression, rather than fitting a regression line, we fit a "S" shaped logistic function that predicts two maximum values (0 or 1).

### 4.3 Logistic Function – Sigmoid Function

- The sigmoid function is a mathematical function that maps anticipated values to probabilities.
- It converts any real value into another value between 0 and 1. The logistic regression value must be between 0 and 1, and it cannot exceed this limit, resulting in a curve similar to the "S" form.
- The S-form curve is also known as the sigmoid function or logistic function.
- In logistic regression, we use the concept of a threshold value, which establishes the likelihood of 0 or 1. For example, numbers above the threshold tend to be 1, whereas those below the threshold likely to be zero.



## 4.4 How does Logistic Regression work?

The logistic regression model uses a sigmoid function to convert the continuous value output of the linear regression function to categorical value output. This function transfers any real-valued collection of independent variables input into a value between 0 and 1. This function is referred to as the logistic function. Let the independent input characteristics be:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e. 0 or 1.

$$Y = \begin{cases} 0 & \text{if Class 1} \\ 1 & \text{if Class 2} \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = \left( \sum_{i=1}^n w_i x_i \right) + b$$

Here  $x_i$  is the  $i$ th observation of X,  $w_i = [w_1, w_2, w_3, \dots, w_m]$  is the weights or Coefficient, and  $b$  is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

whatever we discussed above is the linear regression.

### 4.4.1 Sigmoid Function

Now we use the sigmoid function where the input will be  $z$  and we find the probability between 0 and 1. i.e. predicted  $y$ .

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

As shown above, the figure sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.

- $\sigma(z)$  tends towards 1 as  $z \rightarrow \infty$
- $\sigma(z)$  tends towards 0 as  $z \rightarrow -\infty$
- $\sigma(z)$  is always bounded between 0 and 1

where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z)$$

$$P(y = 0) = 1 - \sigma(z)$$

#### 4.4.2 Logistic Regression Equation

The odd is the ratio of something occurring to something not occurring. it is different from probability as the probability is the ratio of something occurring to everything that could possibly occur. so odd will be:

$$\frac{p(x)}{1 - p(x)} = e^z$$

Applying natural log on odd. then log odd will be:

$$\log \left[ \frac{p(x)}{1 - p(x)} \right] = z$$

$$\log \left[ \frac{p(x)}{1 - p(x)} \right] = w \cdot X + b$$

$$\frac{p(x)}{1 - p(x)} = e^{w \cdot X + b} \quad \dots \text{Exponentiate both sides}$$

$$p(x) = e^{w \cdot X + b} \cdot (1 - p(x))$$

$$p(x) = e^{w \cdot X + b} - e^{w \cdot X + b} \cdot p(x)$$

$$p(x) + e^{w \cdot X + b} \cdot p(x) = e^{w \cdot X + b}$$

$$p(x)(1 + e^{w \cdot X + b}) = e^{w \cdot X + b}$$

$$p(x) = \frac{e^{w \cdot X + b}}{1 + e^{w \cdot X + b}}$$

then the final logistic regression equation will be:

$$p(X; b, w) = \frac{e^{w \cdot X + b}}{1 + e^{w \cdot X + b}} = \frac{1}{1 + e^{-w \cdot X + b}}$$

#### 4.4.3 Likelihood Function for Logistic Regression

The predicted probabilities will be:

- for y=1 The predicted probabilities will be:

$$p(X; b, w) = p(x)$$

- for y=0 The predicted probabilities will be:

$$1 - p(X; b, w) = 1 - p(x)$$

$$L(b, w) = \prod_{i=1}^n p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

Taking natural logs on both sides

$$\begin{aligned} \log(L(b, w)) &= \sum_{i=1}^n y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i)) \\ &= \sum_{i=1}^n y_i \log p(x_i) + \log(1 - p(x_i)) - y_i \log(1 - p(x_i)) \\ &= \sum_{i=1}^n \log(1 - p(x_i)) + \sum_{i=1}^n y_i \log \frac{p(x_i)}{1 - p(x_i)} \\ &= \sum_{i=1}^n -\log 1 - e^{-(w \cdot x_i + b)} + \sum_{i=1}^n y_i (w \cdot x_i + b) \\ &= \sum_{i=1}^n -\log 1 + e^{w \cdot x_i + b} + \sum_{i=1}^n y_i (w \cdot x_i + b) \end{aligned}$$

#### 4.4.4 Gradient of the log-likelihood function

To find the maximum likelihood estimates, we differentiate w.r.t  $w$ ,

$$\begin{aligned} \frac{\partial J(l(b, w))}{\partial w_j} &= - \sum_{i=1}^n \frac{1}{1 + e^{w \cdot x_i + b}} e^{w \cdot x_i + b} x_{ij} + \sum_{i=1}^n y_i x_{ij} \\ &= - \sum_{i=1}^n p(x_i; b, w) x_{ij} + \sum_{i=1}^n y_i x_{ij} \\ &= \sum_{i=1}^n (y_i - p(x_i; b, w)) x_{ij} \end{aligned}$$

# Chapter 5

## Evaluation method

To evaluate classification model, which is Logistics Regression, we use Confusion Matrix, Accuracy, F1-Score, Precision, Recall, Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC).

### 5.0.1 Confusion Matrix

Confusion Matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. The matrix displays the number of instances produced by the model on the test data:

- **True Positive (TP):** The model correctly predicted a positive outcome (the actual outcome was positive).
  - **True Negative (TN):** The model correctly predicted a negative outcome (the actual outcome was negative).
  - **False Positive (FP):** The model incorrectly predicted a positive outcome (the actual outcome was negative). Also known as a Type I error.
  - **False Negative (FN):** The model incorrectly predicted a negative outcome (the actual outcome was positive). Also known as a Type II error.
1. **Accuracy:** Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.
  2. **Precision:** Precision is a measure of how accurate a model's positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.
  3. **Recall:** Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.
  4. **F1-Score:** F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

### 5.0.2 AUC-ROC curve

The AUC-ROC curve, or Area Under the Receiver Operating Characteristic curve, is a graphical representation of the performance of a binary classification model at various classification thresholds. It is commonly used in machine learning to assess the ability of a model to distinguish between two classes, typically the positive class (people who are not obese) and the negative class (people who are obese).

- **Receiver Operating Characteristics (ROC) Curve:** It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.
- **Area Under Curve (AUC) Curve:** It represents the area under the ROC curve and measures the overall performance of the binary classification model.

There are some key points about AUC-ROC curve:

- A high AUC (close to 1) indicates excellent discriminative power. This means the model is effective in distinguishing between the two classes, and its predictions are reliable.
- A low AUC (close to 0) suggests poor performance. In this case, the model struggles to differentiate between the positive and negative classes, and its predictions may not be trustworthy.
- AUC around 0.5 implies that the model is essentially making random guesses. It shows no ability to separate the classes, indicating that the model is not learning any meaningful patterns from the data.

# Chapter 6

## Results

### 6.1 Forward Feature Selection

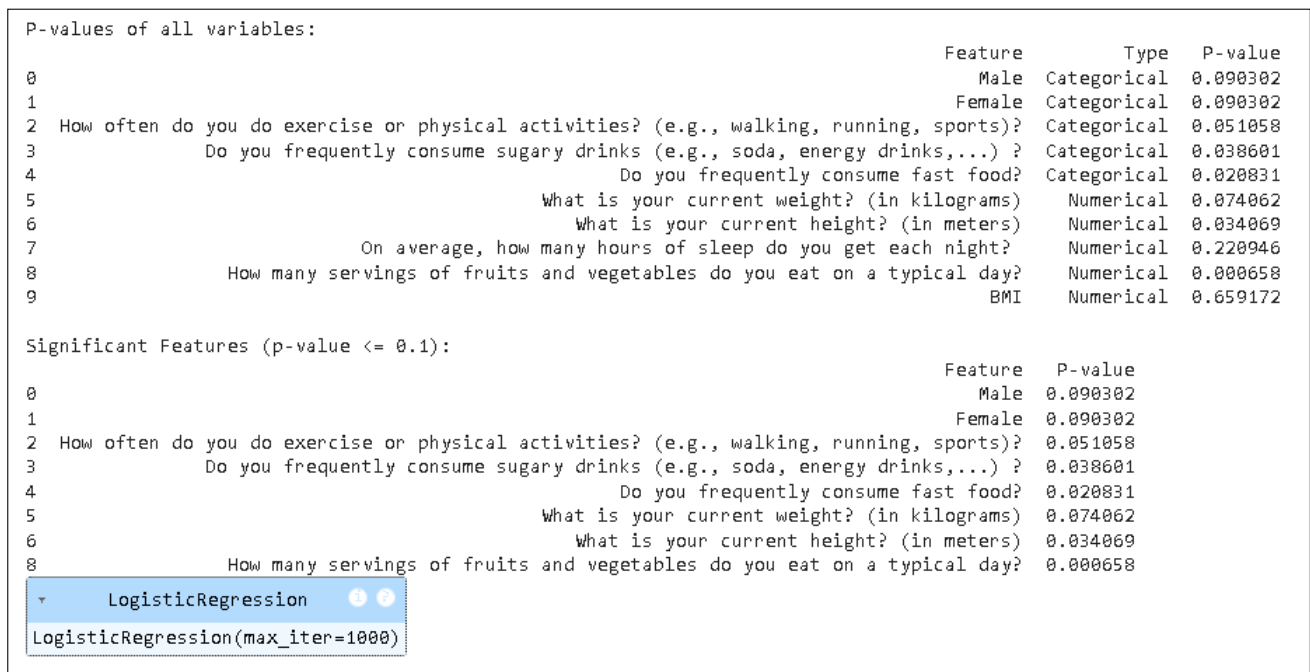


Figure 6.1: Figure Selection

This analysis identifies significant features for a Forward Feature Selection by calculating the p-value of the variables. The threshold we choose is 0.1 because in our project, when the threshold is 0.05, our accuracy score is low due to the fact that only 4 features are below the 0.05, which leads to the low performance of the model. After changing the threshold to 0.1, we only delete the "BMI" variable, which is very related to the label.

### 6.2 Logistics Regression Model

#### 6.2.1 Splitting data

- We split our data into 2 groups. The first one is the training data, which account for 80 percent of total data, and the remaining 20 percent is the test-data.

- We also use the code "random state=42" to ensure the data split is reproducible.
- "stratify = y" ensures that the proportion of the target variable (healthy) in the original dataset is maintained in both the training and testing sets.

### 6.2.2 Training data and Predictions

- We set the maximum iterations is 1000.
- Our method for the model is "fit" method.

## 6.3 Confusion Matrix

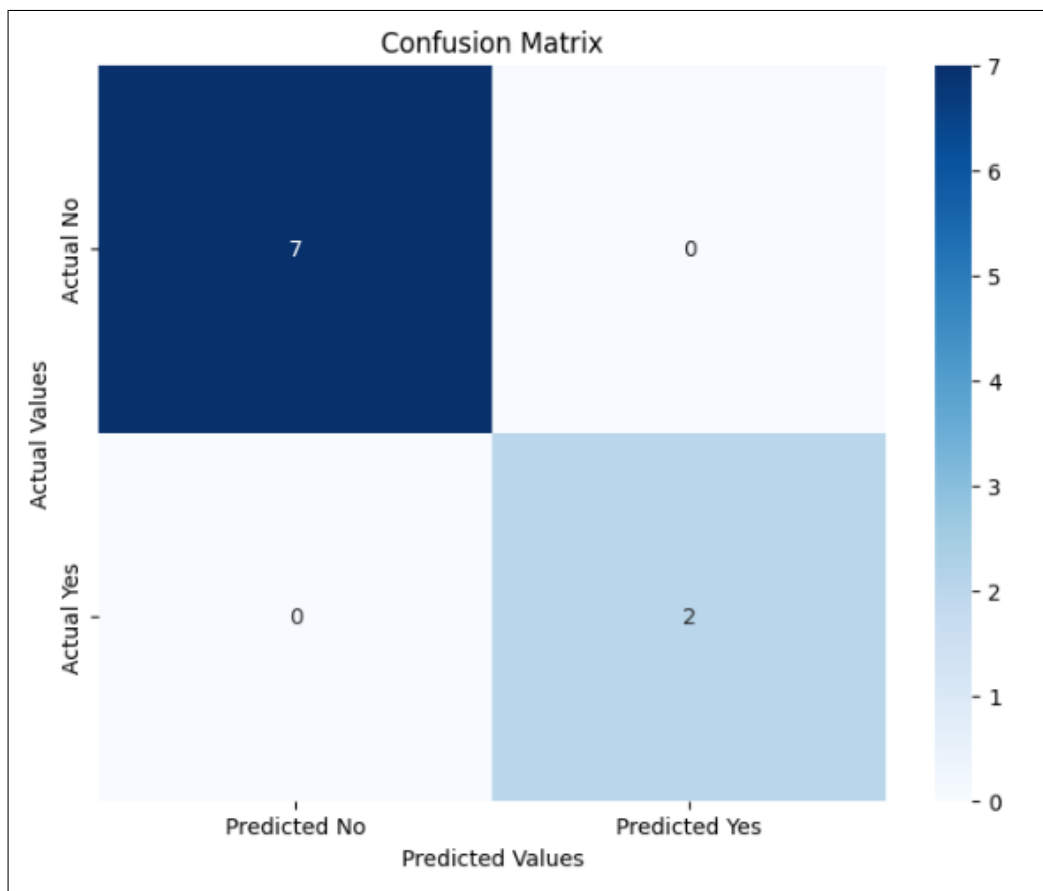


Figure 6.2: Confusion Matrix

- The model performed perfectly on this dataset because it did not produce false positives (FP) or false negatives (FN).
- All predictions (7 "No" and 2 "Yes") matched the actual outcomes, indicating 100 percent accuracy.

- We also calculate the Precision, Recall and F1-Score to evaluate the results, which are 1.0, indicate the best performance of the model.

```
Accuracy of the Logistic Regression model: 1.0
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Figure 6.3: Accuracy of the Logistic Regression model

```
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

Figure 6.4: Evaluation Metric

## 6.4 Receiver Operating Characteristic (ROC) curve

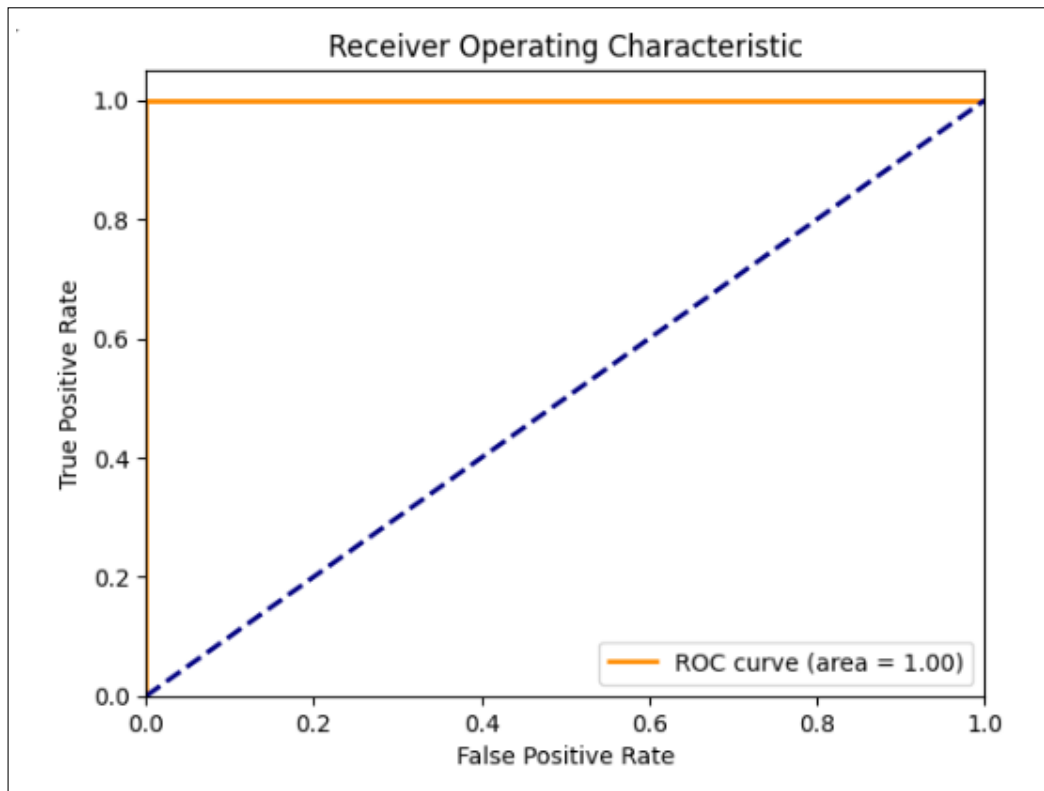


Figure 6.5: Receiver Operating Characteristic Curve

- **The Diagonal Line (Random Guess Line):** This line represents a model with no discrimination ability (random guessing), where the True Positive Rate equals the False Positive Rate.



- **The Orange Line:** This is the ROC curve of the model. The curve being perfectly horizontal at 1.0 TPR shows that the model perfectly distinguishes between positive and negative classes.
- **Area Under the Curve (AUC):** AUC values range from 0.5 (random guessing) to 1.0 (perfect classification). The AUC value is 1.0, which represents a perfect model.
- **Interpretation:** Since the orange ROC curve reaches the top-left corner of the graph and stays at  $\text{TPR} = 1.0$  with  $\text{FPR} = 0$ , the model has 100

# Chapter 7

## Conclusion

This study provided valuable information on the application of logistic regression to predict obesity levels. Using data from health surveys, the logistic regression model highlighted critical variables such as Body Mass Index (BMI), age, and lifestyle behaviors. This analysis not only facilitated accurate predictions, but also underscored the importance of robust pre-processing and data management techniques. The findings have significant implications for the design of effective obesity prevention strategies and the advancement of academic research in health analytics. The study underscores the transformative potential of using logistic regression and data-driven methodologies to tackle obesity. By bridging technical capabilities with pressing healthcare needs, these initiatives pave the way for a healthier and more informed society.

### 7.1 Key Takeaways

The logistic regression analysis in this study effectively demonstrated its ability to predict obesity levels based on data from the health survey. Key insights include:

- **Model Accuracy:** Logistic regression showcased satisfactory predictive accuracy, supported by robust variable selection and preprocessing steps. In particular, variables such as body mass index (BMI), age, and lifestyle behaviors emerged as critical determinants of obesity.
- **Significance of Data Cleaning:** Proper handling of missing, outlier, and irrelevant data significantly improved model performance and reliability.
- **Health Implications:** The findings reinforce the role of healthy dietary habits and regular exercise in mitigating obesity risks. Predictive modeling has proven instrumental in identifying high-risk individuals for targeted health interventions.

In conclusion, Logistics Regression model is particularly suited for binary classification problems (0 or 1), which is because of its simplicity and effectiveness when implementing with small and medium datasets. On the other hand, it is very sensitive to outliers since they can skew the estimated coefficients.

## 7.2 Final Remarks

This project exemplifies the utility of logistic regression in addressing public health challenges such as obesity. By integrating diverse variables, the model provides actionable insights that can inform healthcare strategies and policy decisions. The study highlights:

- The importance of survey-based data collection in acquiring a comprehensive understanding of individual health profiles.
- The adaptability and efficiency of logistic regression in binary classification tasks, making it an ideal choice for predicting health outcomes such as obesity.

While the results are promising, this project also highlights the necessity for continuous model evaluation and data quality improvement. The alignment of technical methodologies with real-world health needs strengthens the credibility and applicability of the findings.

## 7.3 Future Directions

Looking forward, several pathways can expand and enhance this research:

1. **Incorporation of Advanced Models:** Investigate more sophisticated machine learning models such as random forests, support vector machines, or neural networks to compare and potentially improve predictive performance.
2. **Longitudinal Data:** Expanding the dataset to include longitudinal health data can provide deeper insights into trends and causal relationships associated with obesity.
3. **Feature Enrichment:** Introducing additional variables, such as genetic factors, sleep patterns, or stress levels, could improve the model's scope and accuracy.
4. **Population-Specific Models:** Customizing models for specific demographic or geographic groups may yield more tailored and effective health interventions.
5. **Deployment in Real-World Settings:** Collaborating with healthcare organizations to implement and validate the model in clinical environments or community health programs.

# Chapter 8

## References

<https://www.supersurvey.com/Research#Definition>

<https://www.questionpro.com/blog/survey-data-collection/>

<https://www.geeksforgeeks.org/data-cleansing-introduction/>

<https://viblo.asia/p/encoding-categorical-features-in-machine-learning-bJzKm4mw59N>

<https://viblo.asia/p/logistic-regression-bai-toan-co-ban-trong-machine-learning-924lJ4rzKPM>

<https://www.geeksforgeeks.org/understanding-logistic-regression/>

<https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/>

<https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>

<https://www.geeksforgeeks.org/auc-roc-curve/>

# Chapter 9

## Appendix

### 9.1 Survey Questionnaire

The questionnaire includes a comprehensive set of questions that gather demographic information, health metrics, and lifestyle behaviors. These questions are carefully structured to ensure the data's relevance and adequacy for building a predictive model while considering ethical standards and participants' confidentiality.

- **Age:** Captures the participant's age to analyze its influence on obesity levels, as age is often correlated with metabolic and lifestyle changes.



What is your age? \*

Your answer

Figure 9.1: Age Question

- **Gender:** Identifies the participant's gender to study gender-specific patterns in obesity prevalence.



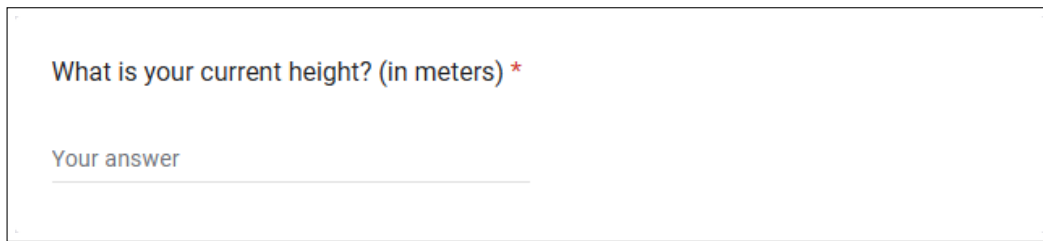
What is your gender \*

☐ Male

☐ Female

Figure 9.2: Gender Question

- **Current Height:** Provides a vital physical metric used to calculate Body Mass Index (BMI), a key variable in the model.



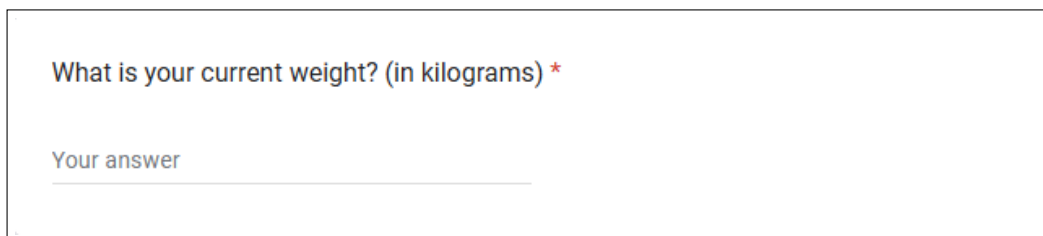
What is your current height? (in meters) \*

Your answer

A screenshot of a web form with a light gray background. The question 'What is your current height? (in meters) \*' is displayed in a dark gray font. Below the question is a text input field with the placeholder text 'Your answer' in a lighter gray font. The form is enclosed in a thin black border.

Figure 9.3: Height Question

- **Current Weight:** Together with height, it enables BMI calculation, a significant determinant in predicting obesity levels.



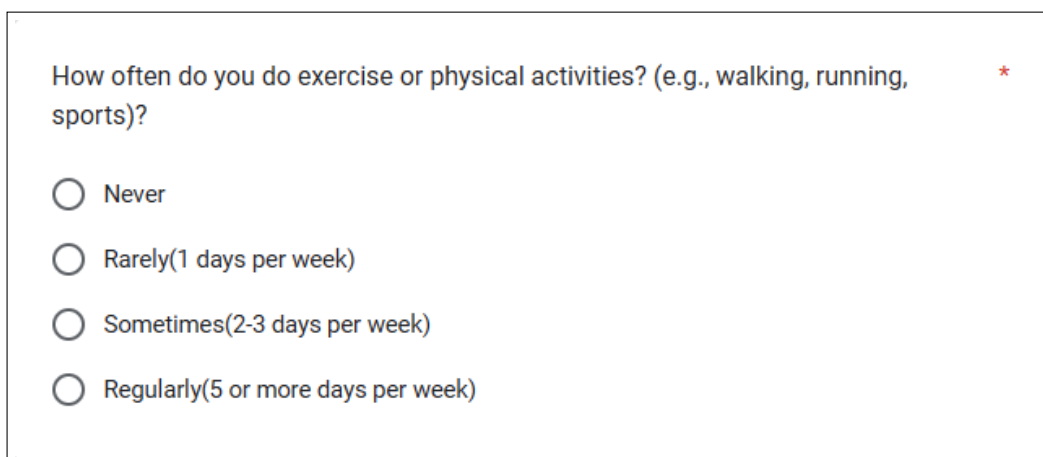
What is your current weight? (in kilograms) \*

Your answer

A screenshot of a web form with a light gray background. The question 'What is your current weight? (in kilograms) \*' is displayed in a dark gray font. Below the question is a text input field with the placeholder text 'Your answer' in a lighter gray font. The form is enclosed in a thin black border.

Figure 9.4: Weight Question

- **Exercise frequency:** Assesses physical exercise frequency, which plays a crucial role in managing body weight and preventing obesity.



How often do you do exercise or physical activities? (e.g., walking, running, sports) \*

☐ Never

☐ Rarely(1 days per week)

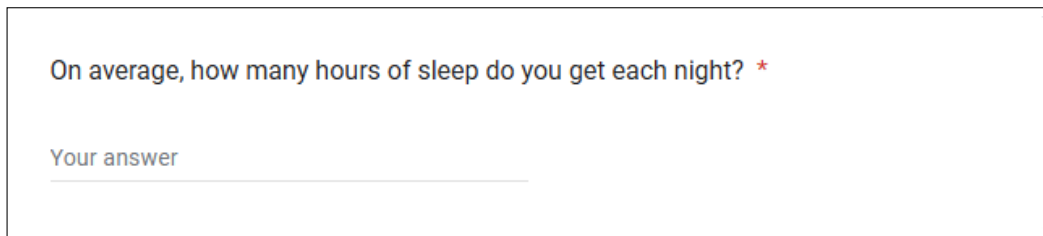
☐ Sometimes(2-3 days per week)

☐ Regularly(5 or more days per week)

A screenshot of a web form with a light gray background. The question 'How often do you do exercise or physical activities? (e.g., walking, running, sports) \*' is displayed in a dark gray font. Below the question are four radio button options: 'Never', 'Rarely(1 days per week)', 'Sometimes(2-3 days per week)', and 'Regularly(5 or more days per week)'. The form is enclosed in a thin black border.

Figure 9.5: Exercise frequency Question

- **Average sleep hours (per night):** Gathers data on sleep duration to assess its relationship with obesity, as inadequate sleep is a potential risk factor.

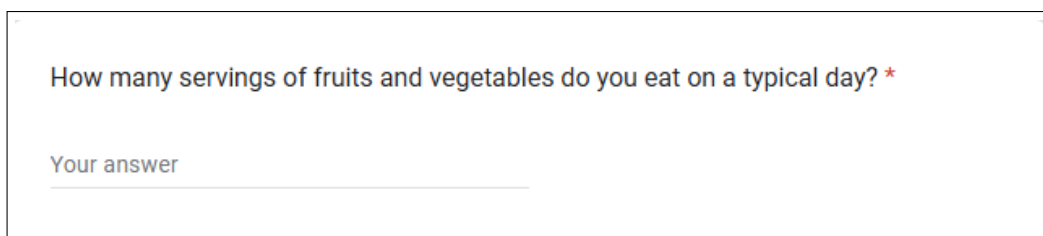


On average, how many hours of sleep do you get each night? \*

Your answer

Figure 9.6: Average Sleep Hours Question

- **Daily servings of fruits and vegetables:** Evaluates dietary habits, as regular consumption of fruits and vegetables is essential for a balanced diet and weight management.

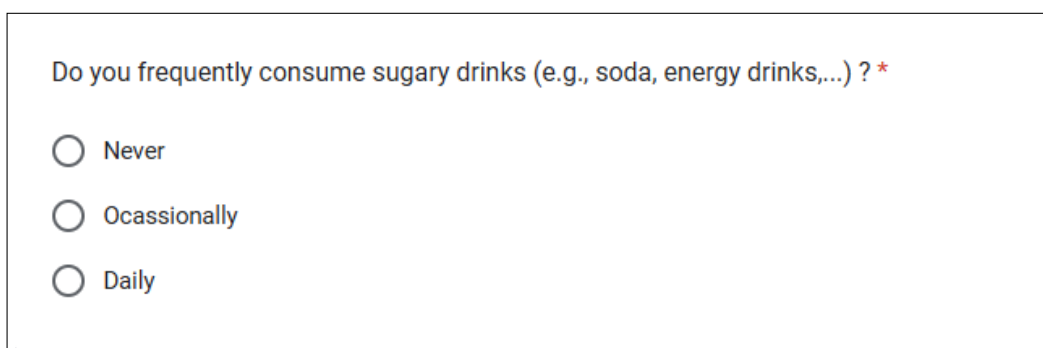


How many servings of fruits and vegetables do you eat on a typical day? \*

Your answer

Figure 9.7: Daily servings of fruits and vegetables Question

- **Frequency of sugary drink consumption:** Measures the intake of sugar-sweetened beverages, which are associated with weight gain and obesity.



Do you frequently consume sugary drinks (e.g., soda, energy drinks,...) ? \*

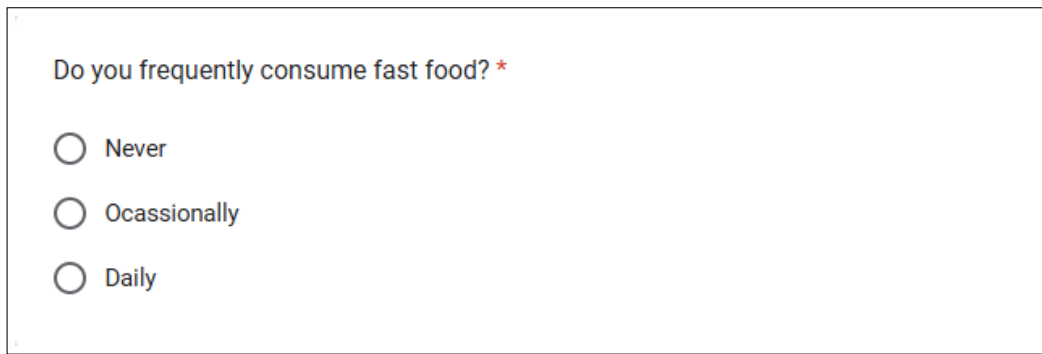
☐ Never

☐ Ocassionally

☐ Daily

Figure 9.8: Frequency of sugary drink consumption Question

- **Frequency of fast consumption:** Examines the frequency of fast food consumption, a known contributor to obesity due to high calorie content.



Do you frequently consume fast food? \*

☐ Never

☐ Ocassionally

☐ Daily

Figure 9.9: Frequency of fast consumption Question

## 9.2 Detailed Statistical Tables

## 9.3 Python code

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.preprocessing import OneHotEncoder
5 from sklearn.preprocessing import OrdinalEncoder
6
7 from google.colab import drive
8 drive.mount('/content/drive')
9
10 df = pd.read_excel("/content/Health Assessment (CÃ¡cu trÃ¡žč lÃ¡žĭi).xlsx"
11                    )
12 df
13 # Extract all the string in the cells in column "What is your current
14   # weight? (in kilograms)"
15 df["What is your current weight? (in kilograms)"] = pd.to_numeric(
16     df["What is your current weight? (in kilograms)"], errors='coerce'
17 ).fillna(
18     df["What is your current weight? (in kilograms)"].str.extract(r"(\d
19     +\.\?\d*)")[0]
20 )
21 # Check the result
```



```

20 df["What is your current weight? (in kilograms)"].value_counts
21
22 #Conver the columns to int
23 df["What is your current weight? (in kilograms)"] = pd.to_numeric(df["
    What is your current weight? (in kilograms)"], errors='coerce').
    fillna(0).astype(int)
24
25 # Extract all the "m" in the cells in column "What is your current
    height? (in meters)"
26 df["What is your current height? (in meters)"] = pd.to_numeric(
27     df["What is your current height? (in meters)"], errors='coerce'
28 ).fillna(
29     df["What is your current height? (in meters)"].str.extract(r"(:(\d
    +)cm(\d*)|(\d+)m(\d*))").apply(
30     lambda x: int(x[0]) * 100 + int(x[1]) if pd.notnull(x[0]) and x[1]
        != '' # cm case
31     else int(x[0]) if pd.notnull(x[0]) # cm case with only the first
        part
32     else int(x[2]) * 100 + int(x[3]) if pd.notnull(x[2]) and x[3] != ''
        # m case
33     else int(x[2]) * 100 if pd.notnull(x[2]) # m case with only the
        first part
34     else np.nan,
35     axis=1
36 )
37 )
38
39 # Remove all the float values: 1.56 into 156
40 df["What is your current height? (in meters)"] = df["What is your
    current height? (in meters)"].apply( lambda x: int(x * 100) if
    isinstance(x, float) and not pd.isna(x) and x < 100 else x)
41
42 #Conver the columns to int
43 df["What is your current height? (in meters)"] = pd.to_numeric(df["What
    is your current height? (in meters)"], errors='coerce').fillna(0).
    astype(int)

```

```

44
45 df
46
47 # Extract all the string in the cells in column "On average, how many
    hours of sleep do you get each night? "
48 df["On average, how many hours of sleep do you get each night? "] = pd.
    to_numeric(
49     df["On average, how many hours of sleep do you get each night? "],
        errors='coerce'
50 ).fillna(
51     df["On average, how many hours of sleep do you get each night? "].
        str.extract(r"(\d+\.? \d*)")[0]
52 )
53
54 #Conver the columns to int
55 df["On average, how many hours of sleep do you get each night? "] = pd.
    to_numeric(df["On average, how many hours of sleep do you get each
        night? "], errors='coerce').fillna(0).astype(int)
56
57 # Extract all the string in the cells in column "How many servings of
    fruits and vegetables do you eat on a typical day?"
58 df["How many servings of fruits and vegetables do you eat on a typical
    day?"] = pd.to_numeric(
59     df["How many servings of fruits and vegetables do you eat on a
        typical day?"], errors='coerce'
60 ).fillna(
61     df["How many servings of fruits and vegetables do you eat on a
        typical day?"].str.extract(r"(\d+\.? \d*)")[0]
62 )
63 # Fill na values with 1, since most answer refering to at least 1
    serving
64 df["How many servings of fruits and vegetables do you eat on a typical
    day?"] = df["How many servings of fruits and vegetables do you eat
        on a typical day?"].fillna(1)
65 #Conver the columns to int

```

```

66 df["How many servings of fruits and vegetables do you eat on a typical
    day?"] = pd.to_numeric(df["How many servings of fruits and
    vegetables do you eat on a typical day?"] , errors='coerce').fillna
    (0).astype(int)
67 # Assuming 100g per serving: 200g -> 2 serving
68 df["How many servings of fruits and vegetables do you eat on a typical
    day?"] = df["How many servings of fruits and vegetables do you eat
    on a typical day?"].apply(
69     lambda x: int(x/100) if isinstance(x, (int)) and not pd.isna(x) and
        x > 100 else x)
70
71 # Encode the Gender columns using pd.dummy
72 # Initialize OneHotEncoder
73 encoder = OneHotEncoder(sparse_output=False)
74
75 # Apply OneHotEncoder to the 'Gender' column
76 gender_encoded = encoder.fit_transform(df[["What is your gender"]])
77
78 # Convert the result into a DataFrame with column names as the
    categories
79 gender_encoded_df = pd.DataFrame(gender_encoded, columns=encoder.
    categories_[0])
80
81 # Join the original DataFrame with the one-hot encoded columns
82 df = df.join(gender_encoded_df)
83
84 # Remove the What is your gender columns
85 df.drop("What is your gender", axis = 1, inplace = True)
86
87 # Encode Other Categorical data
88 # Define the order of categories
89
90 exercise_level = ["Never", "Rarely(1 days per week)", "Sometimes(2-3
    days per week)", "Regularly(5 or more days per week)"]
91 food_consumption_level = ["Never", "Occasionally", "Daily"]
92

```

```

93 # Initialize OrdinalEncoder with the specified categories
94 encoder_exercise = OrdinalEncoder(categories=[exercise_level])
95 encoder_food_consumption = OrdinalEncoder(categories=[
    food_consumption_level])
96
97 # Apply OrdinalEncoder to the columns
98 df["How often do you do exercise or physical activities? (e.g., walking
    , running, sports)?"] = encoder_exercise.fit_transform(df[["How
    often do you do exercise or physical activities? (e.g., walking,
    running, sports)?"]])
99 df["Do you frequently consume sugary drinks (e.g., soda, energy drinks
    ,...) ?"] = encoder_food_consumption.fit_transform(df[["Do you
    frequently consume sugary drinks (e.g., soda, energy drinks,...) ?"
    ]])
100 df["Do you frequently consume fast food?"] = encoder_food_consumption.
    fit_transform(df[["Do you frequently consume fast food?"]])
101
102 # Create a BMI ccolumn
103 df["BMI"] = df["What is your current weight? (in kilograms)"] / ((df["
    What is your current height? (in meters)"] / 100) ** 2)
104
105 # Define label "Healthy"
106 df["healthy"] = df.apply(
107     lambda row: "Yes" if (
108         18.5 <= row["BMI"] <= 24.9 and
109         row["How often do you do exercise or physical activities? (e.g
            ., walking, running, sports)?"] >= 2 and
110         row["Do you frequently consume sugary drinks (e.g., soda,
            energy drinks,...) ?"] <= 1 and
111         row["Do you frequently consume fast food?"] <= 1 and
112         6 <= row["On average, how many hours of sleep do you get each
            night? "] <= 8 and
113         row["How many servings of fruits and vegetables do you eat on a
            typical day?"] >= 2
114     ) else "No",
115     axis=1

```

```

116 )
117 # Encode the label "Healthy" using pd.dummy
118 # Initialize OneHotEncoder
119 encoder = OneHotEncoder(sparse_output=False)
120
121 # Apply OneHotEncoder to the 'Gender' column
122 healthy_encoded = encoder.fit_transform(df[["healthy"]])
123
124 # Convert the result into a DataFrame with column names as the
    categories
125 healthy_encoded_df = pd.DataFrame(healthy_encoded, columns=encoder.
    categories_[0])
126
127 # Join the original DataFrame with the one-hot encoded columns
128 df = df.join(healthy_encoded_df)
129
130 df
131
132 plt.scatter(df.index, df['What is your current height? (in meters)'])
133
134 df[['What is your current height? (in meters)']].apply(pd.Series.
    value_counts, bins = [0,100,150,170,180,190,220,1000000])
135
136 # Filter rows where the height is greater than 220 or less than 125
137 df[(df['What is your current height? (in meters)'] > 220) | (df['What
    is your current height? (in meters)'] < 125)]
138 # Drop row with index 7,10
139 df.drop([7,10],inplace = True)
140 # Replace others outliers with median
141 df.loc[df['What is your current height? (in meters)'] > 220, 'What is
    your current height? (in meters)'] = df['What is your current height
    ? (in meters)'].median()
142
143 df[(df['What is your current height? (in meters)'] > 220) | (df['What
    is your current height? (in meters)'] < 125)]
144

```

```

145 plt.scatter(df.index, df['What is your current weight? (in kilograms)',
146     ])
147
148 # Filter rows where the weight is less than 45kg
149 df[(df['What is your current weight? (in kilograms)'] < 45)]
150 # Drop row with index 12
151 df.drop([12], inplace = True)
152
153 index1 = df[lambda x: x['What is your current height? (in meters)'] >
154     220][:].index
155 df.drop(index1, inplace = True)
156
157 df.describe()
158
159 plt.scatter(df.index, df['On average, how many hours of sleep do you
160     get each night? '])
161
162 index1 = df[lambda x: x['On average, how many hours of sleep do you get
163     each night? '] < 3][:].index
164 df.drop(index1, inplace = True)
165
166 plt.scatter(df.index, df['What is your current height? (in meters)'])
167
168 index1 = df[lambda x: x['What is your current height? (in meters)'] <
169     125][:].index
170 df.drop(index1, inplace = True)
171
172 df
173
174 df.reset_index(drop=True, inplace=True)
175 df.describe()
176
177 import numpy as np
178 import pandas as pd
179 from scipy.stats import chi2_contingency, ttest_ind
180 from sklearn.linear_model import LogisticRegression

```

```

176 from sklearn.model_selection import train_test_split
177
178 # Calculate p-values and store them in a list of dictionaries
179 results = []
180
181 # Categorical features
182 categorical_cols = ["Male", "Female",
183                    "How often do you do exercise or physical
184                    activities? (e.g., walking, running, sports)?",
185                    "Do you frequently consume sugary drinks (e.g.,
186                    soda, energy drinks,...) ?",
187                    "Do you frequently consume fast food?"]
188
189 for col in categorical_cols:
190     contingency_table = pd.crosstab(df[col], df['healthy'])
191     chi2, p, dof, expected = chi2_contingency(contingency_table)
192     results.append({'Feature': col, 'Type': 'Categorical', 'P-value': p
193                   })
194
195 # Numerical features
196 numerical_cols = ["What is your current weight? (in kilograms)",
197                  "What is your current height? (in meters)",
198                  "On average, how many hours of sleep do you get each
199                  night? ",
200                  "How many servings of fruits and vegetables do you
201                  eat on a typical day?", "BMI"]
202
203 for col in numerical_cols:
204     group1 = df[df['healthy'] == 'Yes'][col]
205     group2 = df[df['healthy'] == 'No'][col]
206     t_statistic, p_value = ttest_ind(group1, group2)
207     results.append({'Feature': col, 'Type': 'Numerical', 'P-value':
208                   p_value})
209
210 # Create a DataFrame from the results
211 results_df = pd.DataFrame(results)

```

```

206
207 # Print the p-values of all variables
208 print("P-values of all variables:")
209 print(results_df.to_string())
210
211 # Filter features with p-value <= 0.1
212 significant_features = results_df[results_df['P-value'] <= 0.1][',
    Feature'].tolist()
213
214 # Print the significant features
215 print("\nSignificant Features (p-value <= 0.1):")
216 print(results_df[results_df['Feature'].isin(significant_features)][['
    Feature', 'P-value']].to_string())
217
218 # Choose the target variable
219 target_variable = 'healthy'
220
221 # Prepare data for modeling using selected features
222 X = df[significant_features]
223 y = df[target_variable]
224
225 # Split the data into training and testing sets
226 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42, stratify = y)
227
228 # Initialize and train the Logistic Regression model
229 model = LogisticRegression(max_iter=1000)
230 model.fit(X_train, y_train)
231
232 # ... (rest of your code for model evaluation and prediction) ...
233
234 import pandas as pd
235 from sklearn.model_selection import train_test_split
236 from sklearn.linear_model import LogisticRegression
237 from sklearn.metrics import accuracy_score
238

```



```

239 # Initialize and train the logistic regression model
240 model = LogisticRegression()
241 model.fit(X_train, y_train)
242
243 # Make predictions on the test set
244 y_pred = model.predict(X_test)
245
246 # Evaluate the model
247 accuracy = accuracy_score(y_test, y_pred)
248 print(f"Accuracy of the Logistic Regression model: {accuracy}")
249
250 import matplotlib.pyplot as plt
251 import seaborn as sns
252
253 # Assuming 'df' and other necessary variables are defined as in the
    provided code.
254
255 # ... (your existing code) ...
256
257 # Calculate the correlation matrix
258 correlation_matrix = df.drop(columns=["healthy", "Yes", "No", "Dážděu
    thážli gian"]).corr()
259
260 # Create the heatmap
261 plt.figure(figsize=(12, 10))
262 sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
263 plt.title('Correlation Matrix of Features')
264 plt.show()
265
266 from sklearn.metrics import confusion_matrix
267 import seaborn as sns
268
269 # Assuming y_test and y_pred are already defined from your previous
    code
270 cm = confusion_matrix(y_test, y_pred)
271

```

```

272 plt.figure(figsize=(8, 6))
273 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
274             xticklabels=['Predicted No', 'Predicted Yes'],
275             yticklabels=['Actual No', 'Actual Yes'])
276 plt.xlabel('Predicted Values')
277 plt.ylabel('Actual Values')
278 plt.title('Confusion Matrix')
279 plt.show()
280
281 from sklearn.metrics import roc_curve, auc
282 import matplotlib.pyplot as plt
283
284 # Assuming y_test and y_pred_proba are already defined
285 # Calculate predicted probabilities for the positive class
286 y_pred_proba = model.predict_proba(X_test)[:, 1]
287
288 # Compute ROC curve and AUC (with pos_label specified)
289 fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba, pos_label='Yes')
290 roc_auc = auc(fpr, tpr)
291
292 # Plot the ROC curve
293 plt.figure()
294 plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
    {roc_auc:.2f})')
295 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
296 plt.xlim([0.0, 1.0])
297 plt.ylim([0.0, 1.05])
298 plt.xlabel('False Positive Rate')
299 plt.ylabel('True Positive Rate')
300 plt.title('Receiver Operating Characteristic')
301 plt.legend(loc="lower right")
302 plt.show()
303
304 from sklearn.metrics import precision_score, recall_score, f1_score
305
306 # Assuming y_test and y_pred are already defined

```

```
307 precision = precision_score(y_test, y_pred, pos_label='Yes')
308 recall = recall_score(y_test, y_pred, pos_label='Yes')
309 f1 = f1_score(y_test, y_pred, pos_label='Yes')
310
311 print(f"Precision: {precision}")
312 print(f"Recall: {recall}")
313 print(f"F1 Score: {f1}")
```

# Chapter 10

## Project Contributions Table

Member Name	Contribution
Le Dang Nhat Duong	Feature Selection, Data Mining, Evaluation Model, Report chapter 2,5,6,7
Vuong Thieu Luan	Creating Survey and Slide, Data Preprocessing, Data Cleaning, Report chapter 1,3,7
Nguyen Ngoc Hoang Nam	Creating Survey and Slide, Data Cleaning, Report chapter 2,3,4,7
Cao Thang Long	Creating Survey and Slide, Data Mining, Report chapter 2,3,4,7

Table 10.1: Contribution of Team Members in the Project