COMP5721M: Programming for Data Science

Group project (Coursework 2): Data Analysis Project

Estimation of Obesity Levels Based On Eating Habits and Physical Condition

Give names and emails of group members here:

- *Haojun Jiang*, *sgtj5252*@leeds.ac.uk
- *Yixin Long*, *jnls1593*@leeds.ac.uk
- *Ziheng Chen*, *fjng2803*@leeds.ac.uk
- *Yuhao Shen*, *bthh1050*@leeds.ac.uk

Project Plan

The Data (10 marks)

Obesity is temporarily a prevalent but inevitable issue, drawing a wide concentration in medical and healthy research sections. Among all relevant research topics, finding the potential major factors behind to explain the formation of obesity, and utilizing these factors to conduct tasks like obesity risk estimation and even give analytical medical advices has meaningful implications. For this purpose, we collected data from authorized organizations.

The dataset we used in our project, Estimation of Obesity Levels Based On Eating Habits and Physical Condition, is from UCI Machine Learning Repo (https://archive.ics.uci.edu). This is a single preprocessed file dataset, including realworld data aiming at the estimation of obesity levels in individuals from four countries, based on their eating habits and physical condition.

In the concrete, this dataset was collected through an online platform survey and properly preprocessed by deleting missing data, data normalization, etc. to guarantee the accuracy and reliability. Sequentially, since the initial data distribution calculated with the original survey-collected statistics exists severe class imbalance, the data was properly balanced with Weka and SMOTE by generating up to 77% synthetic records among the eventual 2111 records according to the initial collected data, which helps prevent potential preference on majority class resulted from long-tailed distribution. All these preprocessing details are mentioned at length in [1].

In brief, the dataset consists of 17 attributes (16 features and 1 index of obesity level) and 2111 records (77% generated synthetically, 23% collected directly), provided by CSV format. The variable instructions are shown in the table below.

Relation	Attribute	Role	Type	Description
Basic Condition	Gender	Feature	Categorical	-
	Age	Feature	Continuous	-
	Height	Feature	Continuous	-
	Weight	Feature	Continuous	-
	ObesityHistory	Feature	Binary	Family history with overweight.
Eating Habits	FAVC	Feature	Binary	Do you eat high caloric food frequently?
	FCVC	Feature	Integer	Do you usually eat vegetables in your meals?
	NCP	Feature	Continuous	How many main meals do you have daily?
	CAEC	Feature	Categorical	Do you eat any food between meals?
	SMOKE	Feature	Binary	Do you have the habit of smoking?
	CH2O	Feature	Continuous	How much water do you drink daily?
	CALC	Feature	Categorical	How often do you drink alcohol?
Physical Condition	SCC	Feature	Binary	Do you monitor the calories you eat daily?
	FAF	Feature	Continuous	How often do you have physical activity?

	TUE	Feature	Integer	How much time do you use technological devices such as cell phone, videogames, television, computer and others?
	MTRANS	Feature	Categorical	Which transportation do you usually use?
ObesityLevel	NObesity	Target	Categorical	Obesity level

Finally, all data was labeled and the target class variable ObesityLevel was created with the 7 values below based on equation $BodyMassIndex = \frac{Weight}{height^2}$ and criteria from WHO and Mexican Normative.

- Insufficient Weight
- Normal Weight
- Obesity Type I
- Obesity Type II
- Obesity Type III
- Overweight Level I
- Overweight Level II

Project Aim and Objectives (5 marks)

This project aims to develop a comprehensive understanding of the relationships between various factors and their impact on predicting obesity levels. By analyzing the dataset's features and exploring different modeling techniques, the project seeks to derive meaningful insights and build reasonable predictive models. The objectives of the project are as follows:

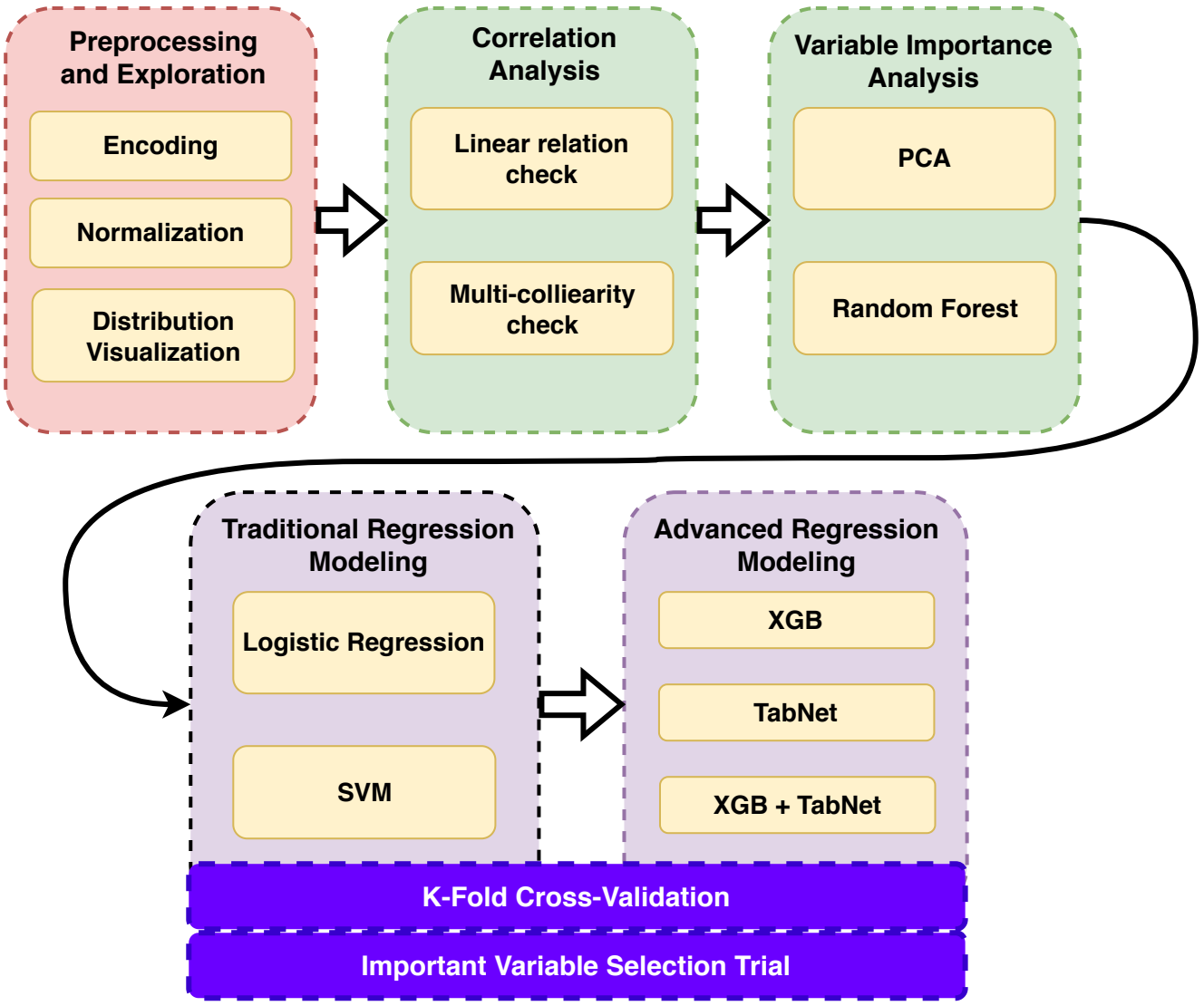
1. Preprocessing and Exploration of Data Distribution: According to the basic property of the dataset, try to explore the potential particularity of the distribution between the 16 features and the target variable. If there are some special distributions between them, find a proper approach to handle them.
2. Correlation Analysis and Variable Importance Measure: Employ the correlation analysis to find the information in the dataset, such as the linear relation between features and target, multi-collinearity between features. Besides, try to utilize rational methods to compare the importance of the features so that dimension reduction may be conducted to optimize the model in the following section.
3. Traditional Model Building: Employing the conclusions about the variable importance, try to utilize the appropriate regression methods to establish a multi-class classification model that can properly handle the continuous and discrete variables. Meanwhile, how to conduct the dimension reduction specifically ought to be discussed.
4. Advanced Modeling: Considering the limitation of traditional regression methods, try to explore advanced methods like Machine Learning and Deep Learning to optimize the accuracy, generalization ability, reliability of the estimation model. Carry out some comparative trials between the different ML/DL methods, together with traditional regression methods, and try to analyze such differences.

Specific Objective(s)

- **Objective 1:**
  Preprocess the dataset with necessary operations, explore potential patterns of the distribution between features and the target variable, and try to extract information from the patterns of the distribution.
- **Objective 2:**
  Select rational methods to analyze the correlation between features and obesity level, and try to draw some conclusions from the results, such as whether the linear relation exists between features and target, or multi-collinearity exists in the features.
- **Objective 3:**
  Utilize an appropriate method to quantify and compare the importance of features, discuss the importance separately in groups of eating habits and physical condition, and conclude practical advice on reducing the risk of obesity accordingly. Quantify the importance of the 16 features in the estimation of obesity levels by certain methods, rank them in descending order, and accordingly conduct a certain important feature selection.
- **Objective 4:**
  Utilize traditional regression models based on the distribution attributes of variables, and establish the obesity level estimation model introduced with certain extent of dimension reduction, so that the model can be properly simplified and optimized.
- **Objective 5:**
  Considering the limitation of traditional regression models, utilize proper machine learning or deep learning models to conduct comparative trials and evaluate the different models with related evaluating indicators.

System Design (5 marks)

Architecture



Processing Modules and Algorithms

- Preprocessing by encoding the categorical variables to numeric and normalization
- Distribution exploration by visualization
- Correlation analysis with Spearman correlation coefficient
- Variable importance analysis by PCA and Random Forest
- K-Fold Cross-Validation
- Traditional multi-class classification regression by Logistic Regression and SVM
- Advanced multi-class classification regression by XGB and TabNet

Program Code (15 marks)

Packages import

Here, all the related packages and modules are simultaneously imported.

Meanwhile, a html table rendering function for outputs in table format is defined to optimize visual display.

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from IPython.display import display, HTML
from xgboost import XGBClassifier
from pytorch_tabnet.tab_model import TabNetClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, label_binarize
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.model_selection import KFold
import torch
import warnings
import os
import sys
import xgboost
from contextlib import contextmanager
from tqdm.notebook import tqdm as tqdm_notebook
from pytorch_tabnet.callbacks import Callback

def display_table_as_html(data, title="Table"):
    """
    Use HTML to render tables.
```

```
    Parameters:
        data (DataFrame): table data to display
        title (str): table title
    """
    style = """
        <style>
        table {border-collapse: collapse; width: 100%; margin: 20px 0; font-size: 14px;}
        th, td {border: 1px solid #ddd; padding: 8px; text-align: left;}
        th {background-color: #f4f4f4;}
        </style>
    """
    html_content = f"<h3>{title}</h3>" + style + data.to_html(index=True, escape=False)
    display(HTML(html_content))
```

Data loading (Explanation of following cell)

Below is a data loading mode, and it shows the basic information of the dataset through `Dataframe` structure in `Pandas`.

In [5]:
```
df = pd.read_csv("data.csv")

# define 3 types of features and target variable
basic_features = ['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight']
eating_habits = ['FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'CALC']
physical_condition = ['SCC', 'FAF', 'TUE', 'MTRANS']
target = 'ObesityLevel'

print("===== Features of the data =====\n", df.columns.values[0:16], "\n")
print("===== Target variable of the data =====\n", df.columns.values[16], "\n")
print("===== Data scale =====\n", df.shape[0], "rows,", df.shape[1], "columns\n")
print("===== Data preview (first 10 rows) =====")
display_table_as_html(df.iloc[:10, :7], "")
display_table_as_html(df.iloc[:10, 8:15], "")
```

```
===== Features of the data =====
 ['Gender' 'Age' 'Height' 'Weight' 'family_history_with_overweight' 'FAVC'
 'FCVC' 'NCP' 'CAEC' 'SMOKE' 'CH2O' 'SCC' 'FAF' 'TUE' 'CALC' 'MTRANS']

===== Target variable of the data =====
 ObesityLevel

===== Data scale =====
 2111 rows, 17 columns

===== Data preview (first 10 rows) =====
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC
0	Female	21.0	1.62	64.0	yes	no	2.0
1	Female	21.0	1.52	56.0	yes	no	3.0
2	Male	23.0	1.80	77.0	yes	no	2.0
3	Male	27.0	1.80	87.0	no	no	3.0
4	Male	22.0	1.78	89.8	no	no	2.0
5	Male	29.0	1.62	53.0	no	yes	2.0
6	Female	23.0	1.50	55.0	yes	yes	3.0
7	Male	22.0	1.64	53.0	no	no	2.0
8	Male	24.0	1.78	64.0	yes	yes	3.0
9	Male	22.0	1.72	68.0	yes	yes	2.0

	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC
0	Sometimes	no	2.0	no	0.0	1.0	no
1	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes
2	Sometimes	no	2.0	no	2.0	1.0	Frequently
3	Sometimes	no	2.0	no	2.0	0.0	Frequently
4	Sometimes	no	2.0	no	0.0	0.0	Sometimes
5	Sometimes	no	2.0	no	0.0	0.0	Sometimes
6	Sometimes	no	2.0	no	1.0	0.0	Sometimes
7	Sometimes	no	2.0	no	3.0	0.0	Sometimes
8	Sometimes	no	2.0	no	1.0	1.0	Frequently
9	Sometimes	no	2.0	no	1.0	1.0	no

Data loading (Comment on previous cell output)

The previous output cell shows the name of 16 features and 1 target variable, and the total number of records is 2111.

These data are ultimately expected to build models using the info extracted from the features to conduct `multi-class classification` tasks in regard to the target variable.

Preprocessing of the data (Explanation of following cell)

Below is the data preprocessing mode, aiming at converting the categorical variables into numeric ones in the ways like binary encoding, weighted one-hot encoding, ordinal encoding. The preprocessed data can sequentially be utilized in the following data analyzing procedures.

In [7]:
```
def preprocess(data):
    """
    Encoding and normalization.

    Parameters:
        data (pd.DataFrame): original data
    Return:
        df_encoded (pd.DataFrame): preprocessed encoded data
    """
    categorical_features = {
        'Gender': 'binary',
        'family_history_with_overweight': 'binary',
        'FAVC': 'binary',
        'FCVC': 'int',
        'NCP': 'int',
        'CAEC': 'ordinal2',
        'SMOKE': 'binary',
        'CH2O': 'int',
        'CALC': 'ordinal2',
        'SCC': 'binary',
        'FAF': 'int',
        'TUE': 'int',
        'MTRANS': 'onehot',
        'ObesityLevel': 'ordinal1'
    }

    # deep copy, avoiding changing the original data
    df_encoded = data.copy()

    # encoding
    for feature, encoding_type in categorical_features.items():
        if encoding_type == 'onehot':
            ordinal_mapping = {'Walking': 0, 'Bike': 1, 'Motorbike': 2,
                               'Automobile': 3, 'Public_Transportation': 4}
            df_encoded[feature] = df_encoded[feature].map(ordinal_mapping)
        elif encoding_type == 'ordinal1':
            ordinal_mapping = {'Insufficient_Weight': 0, 'Normal_Weight': 1,
                               'Obesity_Type_I': 2, 'Obesity_Type_II': 3,
                               'Obesity_Type_III': 4, 'Overweight_Level_I': 5,
                               'Overweight_Level_II': 6}
            df_encoded[feature] = df_encoded[feature].map(ordinal_mapping)
        elif encoding_type == 'ordinal2':
            ordinal_mapping = {'no': 0, 'Sometimes': 1, 'Frequently': 2, 'Always': 3}
            df_encoded[feature] = df_encoded[feature].map(ordinal_mapping)
        elif encoding_type == 'binary':
            ordinal_mapping = {'no': 0, 'yes': 1, 'Male': 1, 'Female': -1}
            df_encoded[feature] = df_encoded[feature].map(ordinal_mapping)
        elif encoding_type == 'int':
            df_encoded[feature] = df_encoded[feature].astype(int)

    # normalization
    features_to_standardize = ['Height', 'Weight', 'Age']
    scaler = StandardScaler()
    df_encoded[features_to_standardize] = scaler.fit_transform(df_encoded[features_to_standardize])

    return df_encoded

df_encoded = preprocess(df)
df1 = df_encoded.loc[(df_encoded["Age"] >= 14.0) & (df_encoded["Age"] < 24.0)]
df2 = df_encoded.loc[(df_encoded["Age"] >= 24.0) & (df_encoded["Age"] <= 64.0)]
print("===== Encoded data preview (first 10 rows) =====")
display_table_as_html(df_encoded.iloc[:10, :7], "")
display_table_as_html(df_encoded.iloc[:10, 8:16], "")
```

```
===== Encoded data preview (first 10 rows) =====
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC
0	-1	-0.522124	-0.875589	-0.862558	1	0	2
1	-1	-0.522124	-1.947599	-1.168077	1	0	3
2	1	-0.206889	1.054029	-0.366090	1	0	2
3	1	0.423582	1.054029	0.015808	0	0	3
4	1	-0.364507	0.839627	0.122740	0	0	2
5	1	0.738817	-0.875589	-1.282647	0	1	2
6	-1	-0.206889	-2.162001	-1.206267	1	1	3
7	1	-0.364507	-0.661187	-1.282647	0	0	2
8	1	-0.049271	0.839627	-0.862558	1	1	3
9	1	-0.364507	0.196421	-0.709799	1	1	2

	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS
0	1	0	2	0	0	1	0	4
1	1	1	3	1	3	0	1	4
2	1	0	2	0	2	1	2	4
3	1	0	2	0	2	0	2	0
4	1	0	2	0	0	0	1	4
5	1	0	2	0	0	0	1	3
6	1	0	2	0	1	0	1	2
7	1	0	2	0	3	0	1	4
8	1	0	2	0	1	1	2	4
9	1	0	2	0	1	1	0	4

Preprocessing of the data (Comment on previous cell output)

The previous output cell shows the preview of the preprocessed data, with encoding and normalization.

Distribution patterns exploration (Explanation of following cell)

Below is the function definition of the data perspective diagrams in the proper form of `histograms and violin plots`, intended to explore the potential distribution patterns and properties inside.

In [8]:
```python
def plot_feature(feature, ax):
    if df[feature].dtype == "object" or len(df[feature].unique()) < 10:
        # categorical
        sns.countplot(data=df, x=feature, hue="ObesityLevel", ax=ax, palette="viridis")
        ax.set_title(f"Distribution of {feature} by Obesity Level")
        ax.set_ylabel("Count")
        ax.tick_params(axis='x', rotation=30)
        for label in ax.get_xticklabels():
            label.set_horizontalalignment('right')  # align the bottom of textbox
    else:
        # continuous
        sns.violinplot(data=df, x="ObesityLevel", hue="ObesityLevel", y=feature, ax=ax, legend=False, palette="viri
        ax.set_title(f"Distribution of {feature} by Obesity Level")
        ax.set_ylabel(feature)
        ax.tick_params(axis='x', rotation=30)
        for label in ax.get_xticklabels():
            label.set_horizontalalignment('right')  # align the bottom of textbox
    ax.set_xlabel("")
```

Data imbalance problem extracted from the distribution properties for features like 'Age' (Explanation of following cell)

Below is the analysis and the visualization module of the data imbalance problem existing in the attribute 'Age', which aims to illustrate the general properties and patterns of such problem, reflecting the quality of the dataset as well.

It is worth mentioning that, in this dataset, the feature Age is solely analyzed for the class imbalance problem because it is a continuous variable valued in certain numeric range, and the distribution in different age groups shows distinct differentiation. That the elder are the minority group in this dataset may lead to little information being learned from the elder group and the model would be weak in generalization on this attribute. The analysis and detailed discussion can be seen in the Object 2 of Project Outcomes section.

In [9]:
```python
def plot_age_group_obesity_distribution(data, age_column, obesity_column, bin_size=2):
    """
```

```
    Plot the obesity level distribution of different age groups.

    Parameters:
        data (pd.DataFrame)
        age_column (str): column names for age divisions
        obesity_column (str): ObesityLevel column names
        bin_size (int): step of age division (default 2)
    """
    # age dividing
    bins = np.arange(data[age_column].min(), data[age_column].max() + bin_size, bin_size)
    age_groups = pd.cut(data[age_column], bins=bins, right=False)

    # conclude distribution of ObesityLevel by age groups
    obesity_distribution = data.groupby(age_groups, observed=True)[obesity_column].value_counts(normalize=False).un

    # plot
    colors = plt.cm.Blues(np.linspace(0.9, 0.3, obesity_distribution.shape[1]))  # color scheme
    obesity_distribution.plot(
        kind='bar',
        stacked=True,
        figsize=(11, 6),
        color=colors,
        width=0.75
    )

    # figure setting
    plt.title(f"Data Imbalance in Age Groups (step={bin_size})", fontsize=14)
    plt.xlabel("Age Groups", fontsize=12)
    plt.ylabel("Count", fontsize=12)
    plt.xticks(rotation=30)
    plt.legend(title="Obesity Level", bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()
    plt.show()
```

Correlation analysis with Spearman coefficient (Explanation of following cell)

Below is the function definition of quantitative correlation analysis between the 2 aspects (Eating Habits and Physical Condition) of features and the target variable (ObesityLevel).

Through correlation analysis, high-related features can be filtered to exploited when modeling, and the relativity can be compared between the two types in order to find the more influential one, which contributes to giving some practical suggestions meaningful to people.

Considering that there are several features are categorical variables, the coefficient used here should be 'Spearman', instead of 'Pearson' traditionally.

In [10]:
```
def plot_spearman_correlation_heatmap():
    """
    Calculate the Spearman corr coefficient and draw a compact heatmap.
    """
    plt.figure(figsize=(9, 9))
    sns.heatmap(
        df_encoded.corr(method='spearman'),
        annot=True,
        fmt=".2f",
        cmap="GnBu",
        cbar=True,
        square=True,
        annot_kws={"fontsize": 8},
        cbar_kws={"shrink": 0.6}
    )
    plt.title('Spearman Correlation Matrix', fontsize=12)
    plt.xticks(fontsize=8, rotation=30, ha='right')  # 调整x轴字体大小和角度
    plt.yticks(fontsize=8)
    plt.tight_layout()
    plt.show()
```

PCA for variable importance comparison (Explanation of following code cell)

Below is the definition of PCA mode, filtering the important features to simplify the model to prevent the over-fitting issue.

Here, the `normalized weighted contribution index` is employed to quantify the importance of each feature, which is

$$NormalizedWeightedContributionFeature_j = \frac{\sum_i ContributionPC_i \cdot Feature_j}{\sum_j \sum_i ContributionPC_i \cdot Feature_j} \times 100\%$$

It is calculated from the contribution index of each Principal Component generated in the PCA procedure, by `adding up the weighted contribution of every feature in the i-th PC vector`.

Additionally, the weighted contribution index are employed to evaluate the importance of the two types of features, Eating Habits and Physical Condition, so that practical suggestions can be concluded for people suffering from obesity.

In [11]:
```python
def perform_pca_analysis_standardized(data, feature_columns):
    """
    Use PCA to perform the importance of the normalized variables, and calculate the normalized weighed contributio

    Parameters:
        data (DataFrame): normalized data
        feature_columns (list): features without height and weight

    Returns:
        pca_components (DataFrame): PC vector
        explained_variance (DataFrame): explained variance of PC
        weighted_contributions (DataFrame): normalized weighted contributions of original features
    """
    # select certain (14) features
    features = data[feature_columns]
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)

    # PCA
    pca = PCA(n_components=len(feature_columns))
    pca.fit(scaled_features)
    components = pca.components_
    explained_variance_ratio = pca.explained_variance_ratio_

    # DataFrame of PC vector
    pca_components = pd.DataFrame(
        components.T,
        columns=[f"PC_{i+1}" for i in range(len(feature_columns))],
        index=feature_columns,
    )

    # PC explained variance table
    explained_variance = pd.DataFrame({
        "Principal Component": [f"PC_{i+1}" for i in range(len(feature_columns))],
        "Explained Variance Ratio": 100 * explained_variance_ratio,
        "Cumulative Variance Ratio": np.cumsum(100 * explained_variance_ratio)
    }).sort_values(by="Explained Variance Ratio", ascending=False)

    # calculate weighted contribution
    weighted_contributions_raw = pca_components.abs().mul(explained_variance_ratio, axis=1).sum(axis=1)
    # normalization to 100% totally
    total_contribution = weighted_contributions_raw.sum()
    weighted_contributions_normalized = 100 * (weighted_contributions_raw / total_contribution)

    # add weighted contributions into DataFrame
    weighted_contributions = pd.DataFrame({
        "Feature": feature_columns,
        "Weighted Contribution (%)": weighted_contributions_normalized,
    }).sort_values(by="Weighted Contribution (%)", ascending=False).reset_index(drop=True)

    weighted_contributions["Cumulative Contribution (%)"] = weighted_contributions["Weighted Contribution (%)"].cum

    return pca_components, explained_variance, weighted_contributions


def plot_pca_results(explained_variance, weighted_contributions):
    """
    Visualize contribution of PC and original features.

    Parameters:
        explained_variance (DataFrame): explained variance of PC
        weighted_contributions (DataFrame): weighted contributions of original features
    """
    # plot contribution of PC
    fig, ax = plt.subplots(1, 1, figsize=(7, 5))
    ax.bar(
        explained_variance["Principal Component"],
        explained_variance["Explained Variance Ratio"],
        color="#00994C"
    )
    ax.set_title("Explained Variance Ratio", fontsize=14)
    ax.set_xlabel("Principal Components")
    ax.set_ylabel("Explained Variance Ratio")
    ax.tick_params(axis='x', rotation=30)
    plt.tight_layout()
    plt.show()

    # plot weighed contribution of original features
    fig, ax = plt.subplots(1, 1, figsize=(7, 5))
    ax.bar(
        weighted_contributions["Feature"],
        weighted_contributions["Weighted Contribution (%)"],
        color="#004C99",
    )
    ax.set_title("Weighted Contributions of Features", fontsize=14)
    ax.set_xlabel("Features")
    ax.set_ylabel("Weighted Contribution (%)")
```

```
    ax.tick_params(axis='x', rotation=30)
    for label in ax.get_xticklabels():
        label.set_horizontalalignment('right')
    plt.tight_layout()
    plt.show()
```

Random Forest for variable importance comparison (Explanation of following code cell)

Below is the function definition of the random forest algorithm in order to quantify the variable importance of the features and compare them by visualization.

In [12]:
```
def compute_random_forest_importance(data, feature_columns, target_column):
    """
    Employ Random Forest to scale the importance of features.

    Parameters:
        data (DataFrame)
        feature_columns (list): selected features
        target_column (str)
    Returns:
        feature_importance (DataFrame)
    """
    X = data[feature_columns]
    y = data[target_column]

    # train Random Forest model
    rf = RandomForestClassifier(n_estimators=100, random_state=9)
    rf.fit(X, y)

    # get variable importance
    importance_values = rf.feature_importances_

    # create variable importance DataFrame
    feature_importance = pd.DataFrame({
        "Feature": feature_columns,
        "Importance (%)": importance_values * 100
    }).sort_values(by="Importance (%)", ascending=False).reset_index(drop=True)

    # cumulative importance
    feature_importance["Cumulative Importance (%)"] = feature_importance["Importance (%)"].cumsum()

    return feature_importance


def plot_feature_importance(feature_importance):
    """
    Visualize the importance of features calculated by Random Forest.

    Parameters:
        feature_importance (DataFrame)
    """
    plt.figure(figsize=(8, 4))
    sns.barplot(x="Importance (%)", y="Feature", hue="Feature", data=feature_importance, palette="viridis")
    plt.title("Feature Importance Based on Random Forest", fontsize=16)
    plt.xlabel("Importance (%)")
    plt.ylabel("Features")
    plt.tight_layout()
    plt.show()
```

Variable importance comparison between Eating Habits and Physical Condition for ObesityLevel (Explanation of following cell)

Below is the function definition to visualize the outcomes of the relativity comparison between Eating habits and Physical condition, intended to illustrate the more effective type resulting in Obesity level. The result helps to make practical suggestions in the Project Outcomes section for obesity people.

In [13]:
```
def compare_feature_categories(weighted_contributions, method='PCA'):
    """
    Compare the proportion of the contribution in two types of features.
    """
    if method == 'PCA':
        eating_habits_contribution = weighted_contributions[weighted_contributions["Feature"].isin(eating_habits)][
        physical_condition_contribution = weighted_contributions[weighted_contributions["Feature"].isin(physical_co
    elif method == 'RandomForest':
        eating_habits_contribution = weighted_contributions[weighted_contributions["Feature"].isin(eating_habits)][
        physical_condition_contribution = weighted_contributions[weighted_contributions["Feature"].isin(physical_co

    # create comparison table
    comparison_df = pd.DataFrame({
        "Category": ["Eating Habits", "Physical Condition"],
        "Total Weighted Contribution (%)": [eating_habits_contribution, physical_condition_contribution]
    })
    print(comparison_df)
```

Non-linear traditional regression modeling (Brief Explanation of following code cell)

Methodology: Logic Regression, SVM

Below is the function definition of the regression modeling for the ObesityLevel estimation.

Considering the properties of the features are discrete and categorical, it is recommended to employ non-linear regression models. After some trials, we find the Logic Regression and the SVM models are the most suitable ones for this task based on this dataset.

In [14]:
```python
def build_model(X_train, y_train, model_type='logistic'):
    scaler = StandardScaler()
    X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns, index=X_train.index)

    if model_type == 'logistic':
        model = LogisticRegression(max_iter=1000, multi_class='ovr', solver='liblinear')
    elif model_type == 'svm':
        model = OneVsRestClassifier(SVC(kernel='linear', probability=True, random_state=9))

    model.fit(X_train_scaled, y_train)
    return model, scaler
```

Model evaluation with K-Fold Cross-Validation and different feature selection scheme on non-linear traditional regression modeling (Brief Explanation of following code cell)

Below are the function definitions of the evaluation of the non-linear traditional regression models. Here, `K-Fold Cross-Validation` is employed to optimize the model in this small-scale dataset. Meanwhile, according to the results fo important variable selection by PCA and Random Forest, models are separately evaluated to compare the performance of the variable importance calculating methods.

In [17]:
```python
def evaluate_model(model, scaler, X_test, y_test, model_type='logistic', ax=None):
    """
    Evaluate the model and return related indexes.
    """
    X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns, index=X_test.index)
    y_pred_proba = model.predict_proba(X_test_scaled)
    y_pred = model.predict(X_test_scaled)

    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, zero_division=1)

    if ax is not None:
        cm = confusion_matrix(y_test, y_pred)
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax, xticklabels=model.classes_, yticklabels=model.cla
        ax.set_title(f"{model_type.capitalize()} - Confusion Matrix")
        ax.set_xlabel('Predicted')
        ax.set_ylabel('True')

    return accuracy, y_pred_proba, report

def plot_multiclass_roc(y_test, y_score, classes, ax):
    """
    Plot ROC Curve of multi-class classification.
    """
    # convert y_test to One-vs-Rest
    y_test_bin = label_binarize(y_test, classes=classes)
    n_classes = y_test_bin.shape[1]  # ensure the correct class num

    # initialize ROC Curve params
    fpr = {}
    tpr = {}
    roc_auc = {}

    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # draw ROC Curve of different classes
    for i in range(n_classes):
        ax.plot(
            fpr[i],
            tpr[i],
            label=f"Class {classes[i]} (AUC = {roc_auc[i]:.2f})"
        )

    ax.plot([0, 1], [0, 1], "k--", label="Random Guess")
    ax.set_xlim([0.0, 1.0])
    ax.set_ylim([0.0, 1.05])
    ax.set_xlabel("False Positive Rate")
    ax.set_ylabel("True Positive Rate")
    ax.set_title(f"Multi-class ROC Curve")
    ax.legend(loc="lower right")

# modified evaluation function with K-Fold Cross-Validation
def evaluate_and_plot_feature_selection(k_folds=5, method='PCA'):
    result1 = perform_pca_analysis_standardized(df_encoded, basic_features + eating_habits + physical_condition)[2]
```

```
    result2 = compute_random_forest_importance(df_encoded, basic_features + eating_habits + physical_condition, tar

    if method == 'PCA':
        feature_importances = result1
        start = 13
    elif method == 'RandomForest':
        feature_importances = result2
        start = 7
    max_features = len(feature_importances)
    accuracy_results_logistic = []
    accuracy_results_svm = []
    best_accuracy_logistic = 0
    best_accuracy_svm = 0
    best_results_logistic = {}
    best_results_svm = {}

    skf = StratifiedKFold(n_splits=k_folds, shuffle=True, random_state=9)

    for num_features in range(start, max_features + 1):
        selected_features = feature_importances[:num_features]
        X = df_encoded[selected_features]
        y = df_encoded['ObesityLevel']

        acc_logistic_fold = []
        acc_svm_fold = []

        for train_index, test_index in skf.split(X, y):
            X_train, X_test = X.iloc[train_index], X.iloc[test_index]
            y_train, y_test = y.iloc[train_index], y.iloc[test_index]

            # Model 1: Logistic
            model_logistic, scaler_logistic = build_model(X_train, y_train, model_type='logistic')
            acc_logistic, y_pred_proba_logistic, report_logistic = evaluate_model(model_logistic, scaler_logistic,
            acc_logistic_fold.append(acc_logistic)

            # Model 2: SVM
            model_svm, scaler_svm = build_model(X_train, y_train, model_type='svm')
            acc_svm, y_pred_proba_svm, report_svm = evaluate_model(model_svm, scaler_svm, X_test, y_test, model_typ
            acc_svm_fold.append(acc_svm)

        avg_acc_logistic = np.mean(acc_logistic_fold)
        avg_acc_svm = np.mean(acc_svm_fold)

        accuracy_results_logistic.append(avg_acc_logistic)
        accuracy_results_svm.append(avg_acc_svm)

        if avg_acc_logistic > best_accuracy_logistic:
            best_accuracy_logistic = avg_acc_logistic
            best_results_logistic = {
                'model': model_logistic,
                'scaler': scaler_logistic,
                'report': report_logistic,
                'accuracy': avg_acc_logistic,
                'X_test': X_test,
                'y_test': y_test,
                'classes': list(set(y_train))
            }

        if avg_acc_svm > best_accuracy_svm:
            best_accuracy_svm = avg_acc_svm
            best_results_svm = {
                'model': model_svm,
                'scaler': scaler_svm,
                'report': report_svm,
                'accuracy': avg_acc_svm,
                'X_test': X_test,
                'y_test': y_test,
                'classes': list(set(y_train))
            }

    # plot accuracy change
    plt.figure(figsize=(8, 4))
    plt.plot(range(start, max_features + 1), accuracy_results_logistic, marker='o', linestyle='-', color='b', label
    plt.plot(range(start, max_features + 1), accuracy_results_svm, marker='o', linestyle='-', color='r', label='SVM
    plt.xlabel('Number of Features Selected')
    plt.ylabel('Average Accuracy')
    plt.xticks(range(start, max_features + 1))
    plt.title(f'Accuracy vs. Number of Selected Features from {method} (K-Fold Cross-Validation)')
    plt.grid(True)
    plt.legend()
    plt.show()

    # output best models' reports
    print(f"\n===== Best Logistic Regression Model Evaluation Based on {method} (Using Top {best_results_logistic['
    print(f"\nLogistic Regression Model Performance:\n{best_results_logistic['report']}")

    print(f"\n===== Best SVM Model Evaluation Based on {method} (Using Top {best_results_svm['model'].n_features_in
```

```
print(f"\nSVM Model Performance:\n{best_results_svm['report']}")

# plot best models' Confusion Matrix and ROC Curves
fig, axes = plt.subplots(2, 2, figsize=(8, 8))

# confusion matrix - Logistic
cm_logistic = confusion_matrix(best_results_logistic['y_test'], best_results_logistic['model'].predict(best_res
sns.heatmap(cm_logistic, annot=True, fmt="d", cmap="Blues", ax=axes[0, 0])
axes[0, 0].set_title("Logistic Regression - Confusion Matrix")
axes[0, 0].set_xlabel("Predicted")
axes[0, 0].set_ylabel("True")

# ROC Curve - Logistic
plot_multiclass_roc(
    best_results_logistic['y_test'],
    best_results_logistic['model'].predict_proba(best_results_logistic['X_test']),
    classes=best_results_logistic['classes'],
    ax=axes[1, 0]
)
axes[1, 0].set_title("Logistic Regression - ROC Curve")
axes[1, 0].legend(loc="best", fontsize=8)

# Confusion matrix - SVM
cm_svm = confusion_matrix(best_results_svm['y_test'], best_results_svm['model'].predict(best_results_svm['X_tes
sns.heatmap(cm_svm, annot=True, fmt="d", cmap="Blues", ax=axes[0, 1])
axes[0, 1].set_title("SVM - Confusion Matrix")
axes[0, 1].set_xlabel("Predicted")
axes[0, 1].set_ylabel("True")
```

```
# ROC Curve - SVM
plot_multiclass_roc(
    best_results_svm['y_test'],
    best_results_svm['model'].predict_proba(best_results_svm['X_test']),
    classes=best_results_svm['classes'],
    ax=axes[1, 1]
)
axes[1, 1].set_title("SVM - ROC Curve")
axes[1, 1].legend(loc="best", fontsize=8)

plt.subplots_adjust(hspace=0.3, wspace=0.2)
plt.show()
```

Advanced regression modeling with ML and DL methods (Brief Explanation of following code cell)

Below is the function definition of using the XGB and TabNet as the model for estimation. To be specific, the XGB, the TabNet, and the combination of XGB and TabNet are employed to evaluate if the performance is better than the traditional regression models. Meanwhile, K-Fold Cross-Validation is utilized, and performance comparison between XGB and TabNet is conducted as well.

In [22]:
```
warnings.filterwarnings("ignore", message=".*Device used : cpu.*")  # suppress specific UserWarnings globally

class TQDMProgressBar(Callback):
    def __init__(self, logs_container):
        self.pbar = None
        self.logs_container = logs_container

    def on_epoch_begin(self, epoch_idx, logs=None):
        if self.pbar is None:
            self.pbar = tqdm_notebook(total=self.trainer.max_epochs, desc="Training", unit="epoch", leave=False)
        self.pbar.update(1)

    def on_epoch_end(self, epoch_idx, logs=None):
        self.pbar.set_postfix(loss=logs['loss'], accuracy=logs.get('val_0_accuracy', None))
        self.logs_container['loss'].append(logs['loss'])
        self.logs_container['accuracy'].append(logs.get('val_0_accuracy', None))

    def on_train_end(self, logs=None):
        if self.pbar is not None:
            self.pbar.close()

@contextmanager
def suppress_stdout():
    with open(os.devnull, "w") as fnull:
        old_stdout = sys.stdout
        sys.stdout = fnull
        try:
            yield
        finally:
            sys.stdout = old_stdout

def build_advanced_model(X_train, y_train, model_type='xgb', X_val=None, y_val=None):
    logs_container = {"loss": [], "accuracy": []}

    if model_type == 'xgb':
        model = XGBClassifier(n_estimators=50, eval_metric='mlogloss', random_state=9, verbosity=0)
        if X_val is not None and y_val is not None:
            eval_set = [(X_val, y_val)]
```

```
            with suppress_stdout():
                model.fit(X_train, y_train, eval_set=eval_set, verbose=True)
            logs_container["loss"] = model.evals_result()["validation_0"]["mlogloss"]
            for epoch in range(len(logs_container["loss"])):
                y_val_pred = model.get_booster().predict(
                    xgboost.DMatrix(X_val),
                    iteration_range=(0, epoch + 1)
                )
                y_val_pred = [np.argmax(prob) for prob in y_val_pred]
                acc = accuracy_score(y_val, y_val_pred)
                logs_container["accuracy"].append(acc)
        else:
            model.fit(X_train, y_train)
        return model, logs_container

    elif model_type == 'tabnet':
        X_train_values = X_train.values
        y_train_values = y_train.values
        if X_val is not None and y_val is not None:
            X_val_values = X_val.values
            y_val_values = y_val.values
            model = TabNetClassifier(device_name='cuda' if torch.cuda.is_available() else 'cpu', verbose=0)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore", UserWarning)
                model.fit(X_train_values, y_train_values, eval_set=[(X_val_values, y_val_values)],
                          max_epochs=500, patience=50, batch_size=1024, virtual_batch_size=128,
                          callbacks=[TQDMProgressBar(logs_container)])
        else:
            model = TabNetClassifier(device_name='cuda' if torch.cuda.is_available() else 'cpu', verbose=0)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore", UserWarning)
                model.fit(X_train_values, y_train_values,
                          max_epochs=500, patience=50, batch_size=1024, virtual_batch_size=128,
                          callbacks=[TQDMProgressBar(logs_container)])
        return model, logs_container

    elif model_type == 'xgb_tabnet':
        xgb = XGBClassifier(n_estimators=50, eval_metric='mlogloss', random_state=9, verbosity=0)
        if X_val is not None and y_val is not None:
            eval_set = [(X_val, y_val)]
            xgb.fit(X_train, y_train, eval_set=eval_set, verbose=False)
        else:
            xgb.fit(X_train, y_train)
        X_train_transformed = xgb.apply(X_train)
        if X_val is not None:
            X_val_transformed = xgb.apply(X_val)
            model = TabNetClassifier(device_name='cuda' if torch.cuda.is_available() else 'cpu', verbose=0)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore", UserWarning)
                model.fit(X_train_transformed, y_train.values,
                          eval_set=[(X_val_transformed, y_val.values)],
                          max_epochs=500, patience=50, batch_size=1024, virtual_batch_size=128,
                          callbacks=[TQDMProgressBar(logs_container)])
        else:
            model = TabNetClassifier(device_name='cuda' if torch.cuda.is_available() else 'cpu', verbose=0)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore", UserWarning)
                model.fit(X_train_transformed, y_train.values,
                          max_epochs=500, patience=50, batch_size=1024, virtual_batch_size=128,
                          callbacks=[TQDMProgressBar(logs_container)])
        return (xgb, model), logs_container

# K-fold evaluation function
def evaluate_models_kfold(models, X, y, n_splits=5):
    kfold = KFold(n_splits=n_splits, shuffle=True, random_state=9)
    results = {name: [] for name in models.keys()}
    logs_data = {name: {"loss": [], "accuracy": []} for name in models.keys()}

    for train_idx, test_idx in kfold.split(X):
        X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
        y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

        # Split training data further into train and validation sets for early stopping
        X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state

        for name, model_func in models.items():
            model, logs = model_func(X_train_split, y_train_split, X_val, y_val)

            if name == 'TabNet':
                y_pred = model.predict(X_test.values)
            elif name == 'XGB+TabNet':
                xgb, tabnet = model
                X_test_transformed = xgb.apply(X_test)
                y_pred = tabnet.predict(X_test_transformed)
            else:
                y_pred = model.predict(X_test)
```

```
            acc = accuracy_score(y_test, y_pred)
            report = classification_report(y_test, y_pred, output_dict=True, zero_division=1)
            results[name].append((acc, report))

            logs_data[name]["loss"].append(logs["loss"])
            logs_data[name]["accuracy"].append(logs["accuracy"])

    # Average results over all folds
    for name in results:
        avg_acc = np.mean([result[0] for result in results[name]])
        print(f"{name} Model Average Accuracy: {avg_acc:.2f}")
        avg_report = pd.concat([pd.DataFrame(result[1]) for result in results[name]]).groupby(level=0).mean()
        print(f"Classification Report for {name} Model (Averaged over {n_splits} folds):")
        print(avg_report.transpose())

    # visualize training process
    fig, axes = plt.subplots(2, 3, figsize=(18, 10))
    for idx, (name, logs) in enumerate(logs_data.items()):
        ax_loss = axes[0, idx]
        ax_acc = axes[1, idx]

        # plot log of every fold
        for fold_idx, fold_loss in enumerate(logs["loss"]):
            ax_loss.plot(fold_loss, label=f"Fold {fold_idx + 1}", alpha=0.6)
        for fold_idx, fold_acc in enumerate(logs["accuracy"]):
            ax_acc.plot(fold_acc, label=f"Fold {fold_idx + 1}", alpha=0.6)

        ax_loss.set_title(f"{name} - Loss")
        ax_loss.set_xlabel("Epochs")
        ax_loss.set_ylabel("Loss")
        ax_loss.legend(fontsize=8)

        ax_acc.set_title(f"{name} - Accuracy")
        ax_acc.set_xlabel("Epochs")
        ax_acc.set_ylabel("Accuracy")
        ax_acc.legend(fontsize=8)

    plt.tight_layout()
    plt.show()

# defining models with updated lambda functions for validation set usage
models = {
    'XGBoost': lambda X_train, y_train, X_val, y_val: build_advanced_model(X_train, y_train, model_type='xgb', X_va
    'TabNet': lambda X_train, y_train, X_val, y_val: build_advanced_model(X_train, y_train, model_type='tabnet', X_
    'XGB+TabNet': lambda X_train, y_train, X_val, y_val: build_advanced_model(X_train, y_train, model_type='xgb_tab
}
```

Project Outcome (10 + 10 marks)

Overview of Results

In this project, certain procedures are organized to handle all the set objectives and the outcomes can be concluded as follows.

For Objective 1, histograms and violin plots are employed to visualize the `distribution patterns of all the features`, and there is a `data imbalance issue exists in features like 'Age'`.

For Objective 2, `Spearman correlation coefficient shows weak linear relation between each feature and the target`, and `no multi-collinearity exists between all the features`. The features perform overall good for the estimation modeling.

For Objective 3, two methods, PCA and Random Forest, are employed to quantify the variable importance and provide reference for the important variable selection when modeling afterwards. `The PCA performs almost equally important for all the features, while the result of Random Forest gives a distinct different importance among different features`.

For Objective 4, two traditional regression models, Logistic Regression and SVM, are utilized to conduct the multi-class classification task. Specifically, `K-Fold Cross-Validation is utilized` to optimize the robustness and reliability of the models, and performance comparison between Logistic Regression and SVM is conducted to discuss the better model. Besides, the conclusion of the variable importance is used to test the best number of features ought to be included into the model, so that the best performance could be reached. Finally, `Random Forest is found as the better method for important variable selection, and the Logistic Regression performs slightly better than SVM in this task`.

For Objective 5, similar process of the modeling and evaluation are conducted as Objective 4, but more advanced models are employed to fit the data better. Since the ML/DL models rely on large scale of data, the important feature selection would not be discussed here that all the features are normally add into the dateset used to establish the estimation model, and the `K-Fold Cross-Validation is also utilized`. Eventually, `the advanced performs better than all the traditional regression models by expanding the max epoch, and the loss function correctly and continuously converges in the process`.

Objective 1: Preprocessing and Exploration of Data Distribution

Explanation of Results

Histograms and violin plots are employed to visualize the distribution patterns of all the features, discrete and continuous respectively, versus obesity level. By observing the 16 plots, the distribution of all the features can be clearly displayed, and some properties can be extracted from them.

For instance, `the distribution of Height vs ObesityLevel (line 1, column 3) looks uniform and symmetrical`, which illustrates `the feature Height has good statistical characteristics and the model established based on it can be relatively trusted`.

For another example, `the distribution of MTRANS vs ObesityLevel (last line, last column)` shows that `most people in the dataset choose public transportation and automobile as the main way to going out`. Although the distribution of the approaches of going out is absolutely different, it still includes such information that `it may be just the routine which the investigated people follow in such region, and the way of going out seems to have little impact on the obesity level`.

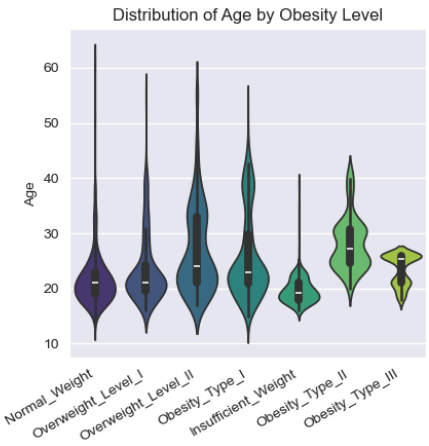
However, `there is a data imbalance issue exists in feature 'Age'`. 'Age' is the only continuous attribute, and it performs that the youth account for the majority in the dataset. It is the phenomenon called data imbalance, causing more estimation errors when it comes to the aged because the information extracted from the dataset is not enough to conclude the properties of the elderly people.

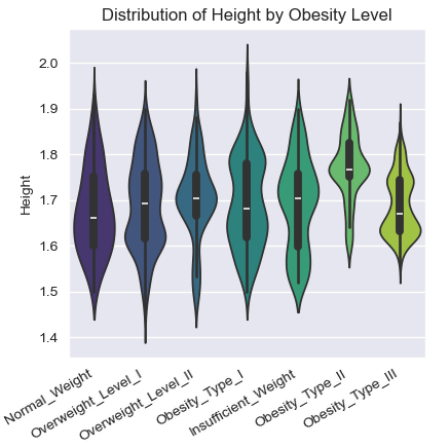
It is worth mentioning that, the other features are not considered to separately modeling because they all have only discrete categories values instead of continuous range of values. So, even data imbalance occurs, information can be extracted from the categories into the traditional regression models.

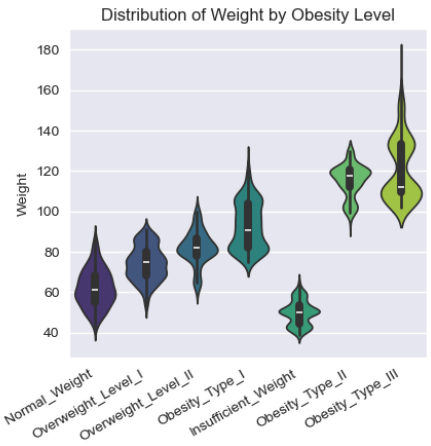
Visualisation

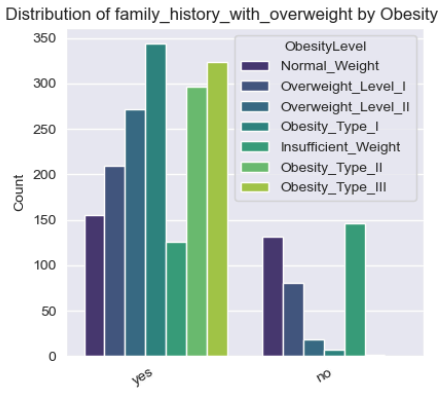
The following multi-frame including histograms and violin plots gives a vivid r`epresentation of the distribution of all features` in regard to obesity level. The sole histogram illustrates the `data imbalance in 'Age'` in detail by visualizing it as this form of plot.

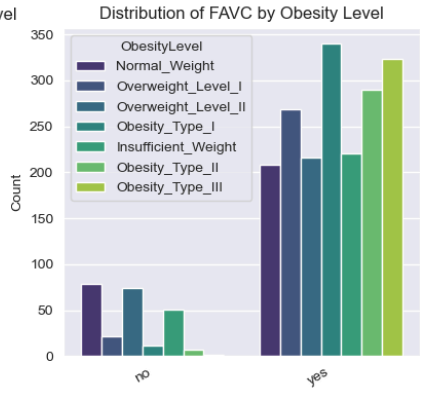
In [87]:
```
# distribution patters exploration
fig, axes = plt.subplots(4, 4, figsize=(18, 18))
axes = axes.flatten()
features = df.columns[:-1]
for i, feature in enumerate(features):
    plot_feature(feature, axes[i])
plt.tight_layout()
plt.subplots_adjust(hspace=0.4, wspace=0.3)
plt.show()

# data imbalance in 'Age'
plot_age_group_obesity_distribution(df, age_column="Age", obesity_column="ObesityLevel")
```

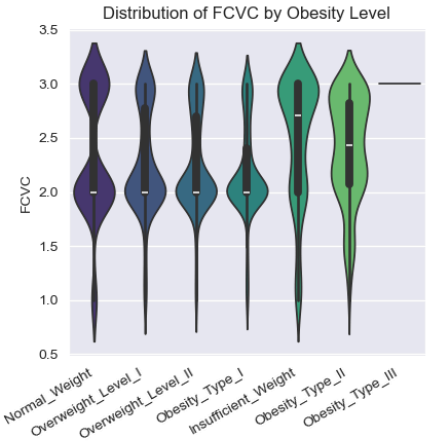


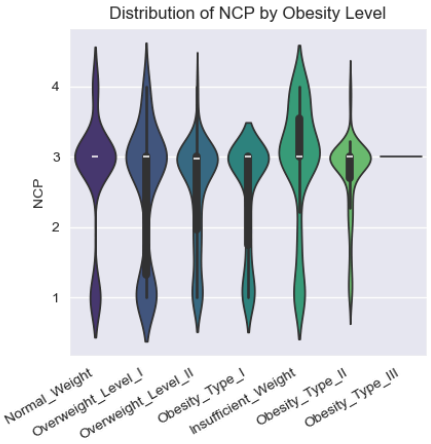


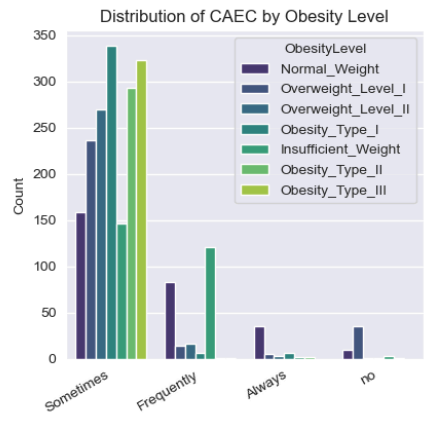


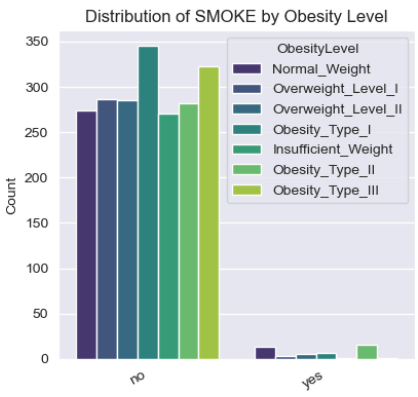


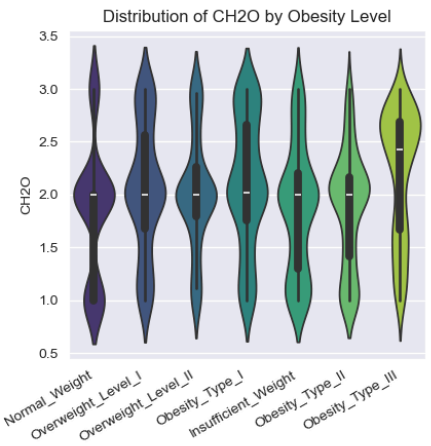


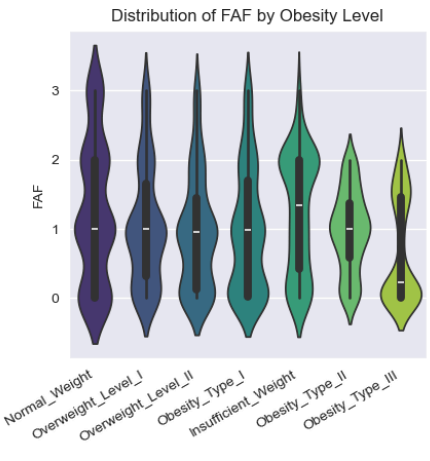


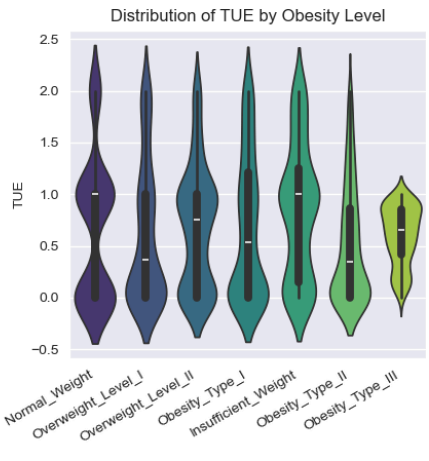


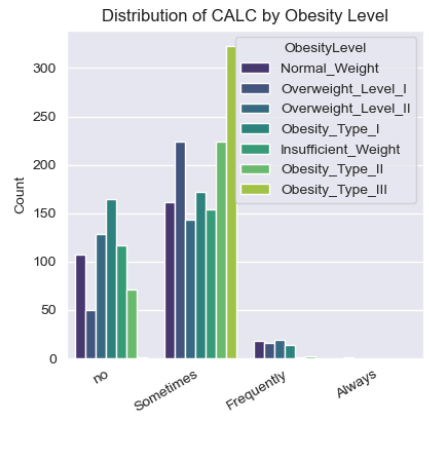


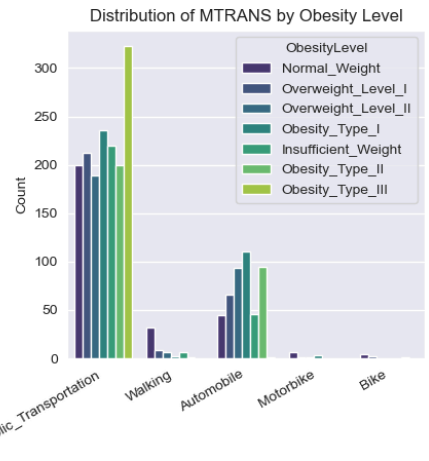


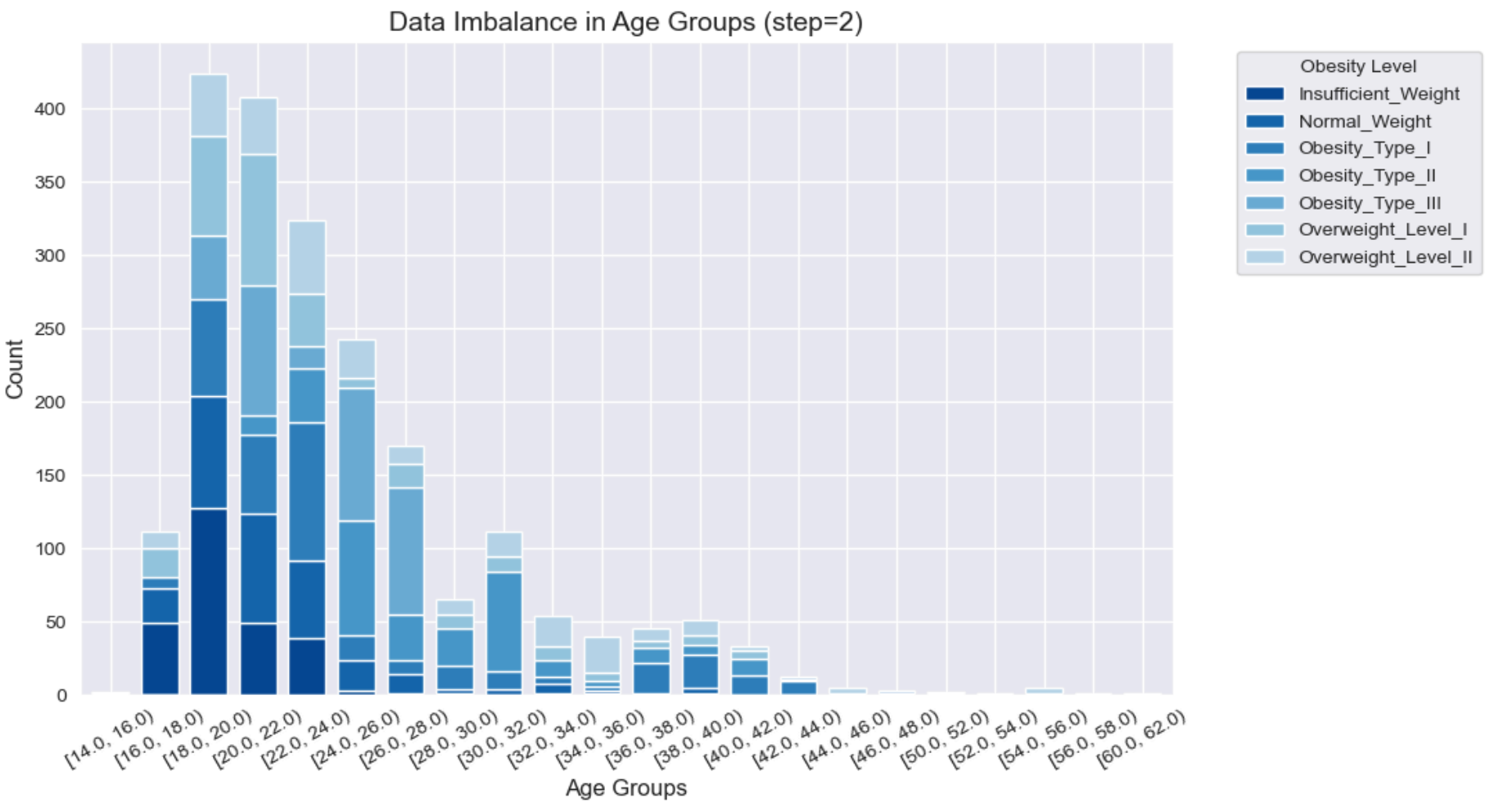












Objective 2: Correlation Analysis and Variable Importance Measure

Explanation of Results

Normally, the following rules are utilized to interpret the correlation coefficient:

- `Strong correlation between the feature and the target variable`: Look for significant features: if the correlation coefficient between a feature and the target variable is high (either positive or negative), then this feature may be a significant feature for predicting the target. For example, if the correlation coefficient between a feature and “ObesityLevel” is close to 1 or -1, then it has a greater impact on predicting obesity level.
- `Correlation between features`:
  - Multicollinearity: If the correlation coefficient between two features is very high (close to 1 or -1), then they may have strong covariance. In a model, the covariance of features may lead to model instability, affecting the interpretability and generalization ability of the model.
  - Redundant features: high correlation between features may indicate that one of the features is redundant and may be considered for removal to simplify the model.
- `Clues to discovering features`: The matrix of correlation coefficients allows us to find out the relationship between some features and others. For example, certain dietary habit features are significantly and positively correlated with obesity class, which means that dietary habits have a greater impact on obesity class, and the impact of these dietary habits can be further analyzed.

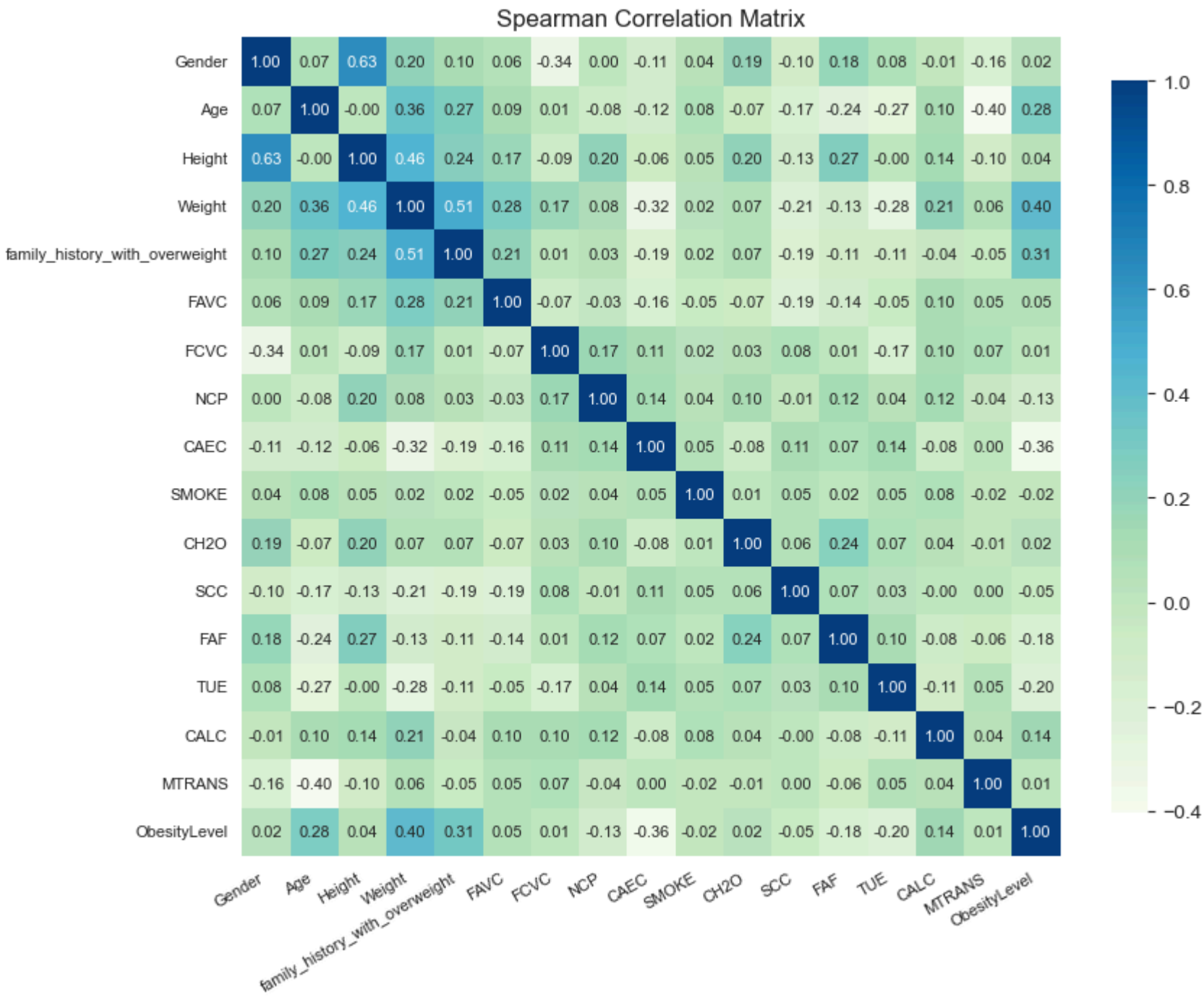
From the visualization results, it can be concluded that:

1. `The most mutually related features are 'Gender' and 'Height' with the coefficient of 0.63`. It should be supposed that `the two feature have certain linear relation that, the older, the higher`. This argument is decent and rational because it is the universal law in real life, while some may stop growing in height as they age.
2. `No multi-collinearity exists among all the features` in that all the coefficient are not close to 1.0.
3. `The features and the target shows weak linear relation` because most of the coefficient are less than 0.5. Considering most of the features are categorical types, this conclusion is reasonable to some extent.

Visualisation

Below is the `heatmap` of the `Spearman correlation coefficient` matrix.

In [36]:
```
plot_spearman_correlation_heatmap()
```



Objective 3: Variable Importance Comparison and Selection

Explanation of Results

It is significant that the model has the proper number of features to prevent overfitting problem and include appropriate amount of information can be extracted from certain features. Therefore, quantifying the variable importance and filtering the vital features are necessary before modeling.

In this part, PCA and Random Forest methods are employed to quantify the variable importance. Specifically, the `weighed contribution index` is used to calculate the importance in PCA with the formula as follows.

$$NormalizedWeightedContributionFeature_j = \frac{\sum_i ContributionPC_i \cdot Feature_j}{\sum_j \sum_i ContributionPC_i \cdot Feature_j} \times 100\%$$

Meanwhile, the theory of the `purity` when the decision grows helps quantify the importance of each feature.

Methodology 1: PCA

From the three tables listed below, it includes:

- the vector of PC
- the explained variance of PC
- the normalized weighed contribution index

Outcomes: The last two tables are plotted as the first two figures, illustrating that:

1. the variable importance for all the features are almost symmetrical and uniform with each other --- all the features show similar importance.
2. `FAVC, TUE, CAEC, Gender, NCP, CH2O, CALC, family_history_with_overweight, FAF, Height, SCC, SMOKE, FCVC` are the main important features if the threshold of cumulative contribution ratio is set to 80%.
3. `Eating Habits` and `Physical Condition` takes up 45.38% and 24.76% respectively for the contribution ratio, which means `Eating Habits` is relatively more important to affecting the obesity. So the people suffering from it can improve their eating habits to gain better efficiency when decreasing weight.

Methodology 2: Random Forest

Outcomes: From the subsequent tables and the corresponding histogram, it can be concluded that:

1. `Weight, Height, Age, FCVC, Gender, family_history_with_overweight, CALC are the main important features if the threshold of cumulative contribution ratio is set to 80%`, where `Weight plays the extremely significant role (36% solely)`.
2. Since there are fewer features accounting for the major important ones than that of PCA, it can be concluded `the Random Forest is able to filter more representative features`.
3. `Eating Habits` and `Physical Condition` takes up `45.38% and 24.76% respectively` for the contribution ratio, which means `Eating Habits is relatively more important to affecting the obesity`. So `people suffering from it can improve their eating habits to gain better efficiency when decreasing weight`.

Visualisation

Below are two code cells of visualization, the first part belongs to PCA, and the other one belongs to Random Forest, including the statistical table of the results and the visualization of the variable importance.

In [37]:
```
## PCA for variable importance

pca_components, explained_variance, weighted_contributions = perform_pca_analysis_standardized(df_encoded, basic_fe

# PCA results
display_table_as_html(pca_components.iloc[: , :8 ], "Principal Component Vectors")
display_table_as_html(pca_components.iloc[: , list(range(8, 16)) ], "")
display_table_as_html(explained_variance, "Explained Variance Ratio")
display_table_as_html(weighted_contributions, "Weighted Contributions")

# visualization
plot_pca_results(explained_variance, weighted_contributions)

# compare importance between Eating Habits and Physical Condition with PCA
compare_feature_categories(weighted_contributions)
```

Principal Component Vectors

	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6	PC_7	PC_8
Gender	-0.294874	0.427295	-0.277097	0.072656	0.047880	0.129610	-0.126938	0.249105
Age	-0.198879	-0.198585	-0.021751	0.640888	0.059817	0.009737	-0.039935	-0.056332
Height	-0.419573	0.390413	0.036727	-0.081572	0.082988	-0.008606	-0.111744	0.236231
Weight	-0.510335	-0.112808	0.214797	-0.073100	-0.074869	-0.011263	0.100283	0.068310
family_history_with_overweight	-0.295248	-0.092766	0.004225	0.046929	-0.074033	-0.281102	0.428133	-0.026475
FAVC	-0.287247	-0.156256	-0.173815	-0.256984	0.205412	-0.060328	-0.085106	-0.164759
FCVC	0.019139	-0.147116	0.620969	-0.023424	-0.168982	-0.184250	0.120246	-0.056784
NCP	-0.089866	0.189651	0.423520	-0.112905	0.260160	-0.310222	-0.190144	-0.003949
CAEC	0.250484	0.144896	0.200735	0.051427	0.422539	-0.331441	-0.056827	0.230909
SMOKE	-0.023974	0.078051	0.151609	0.195855	0.480825	0.385459	0.602386	0.120900
CH2O	-0.071776	0.355569	0.145074	-0.060763	-0.411706	0.175092	0.236293	-0.414574
CALC	-0.147101	-0.045348	0.312226	-0.092168	0.277012	0.535621	-0.407915	-0.372727
SCC	0.245069	0.091240	0.211813	0.030315	-0.177285	0.385884	0.083548	0.365639
FAF	0.076044	0.473632	0.121448	0.010097	-0.230014	-0.084876	-0.031197	0.052945
TUE	0.183638	0.297824	-0.190942	-0.158788	0.310969	-0.135690	0.294025	-0.493993
MTRANS	-0.011300	-0.212788	-0.046601	-0.641109	0.026755	0.139858	0.189373	0.289756

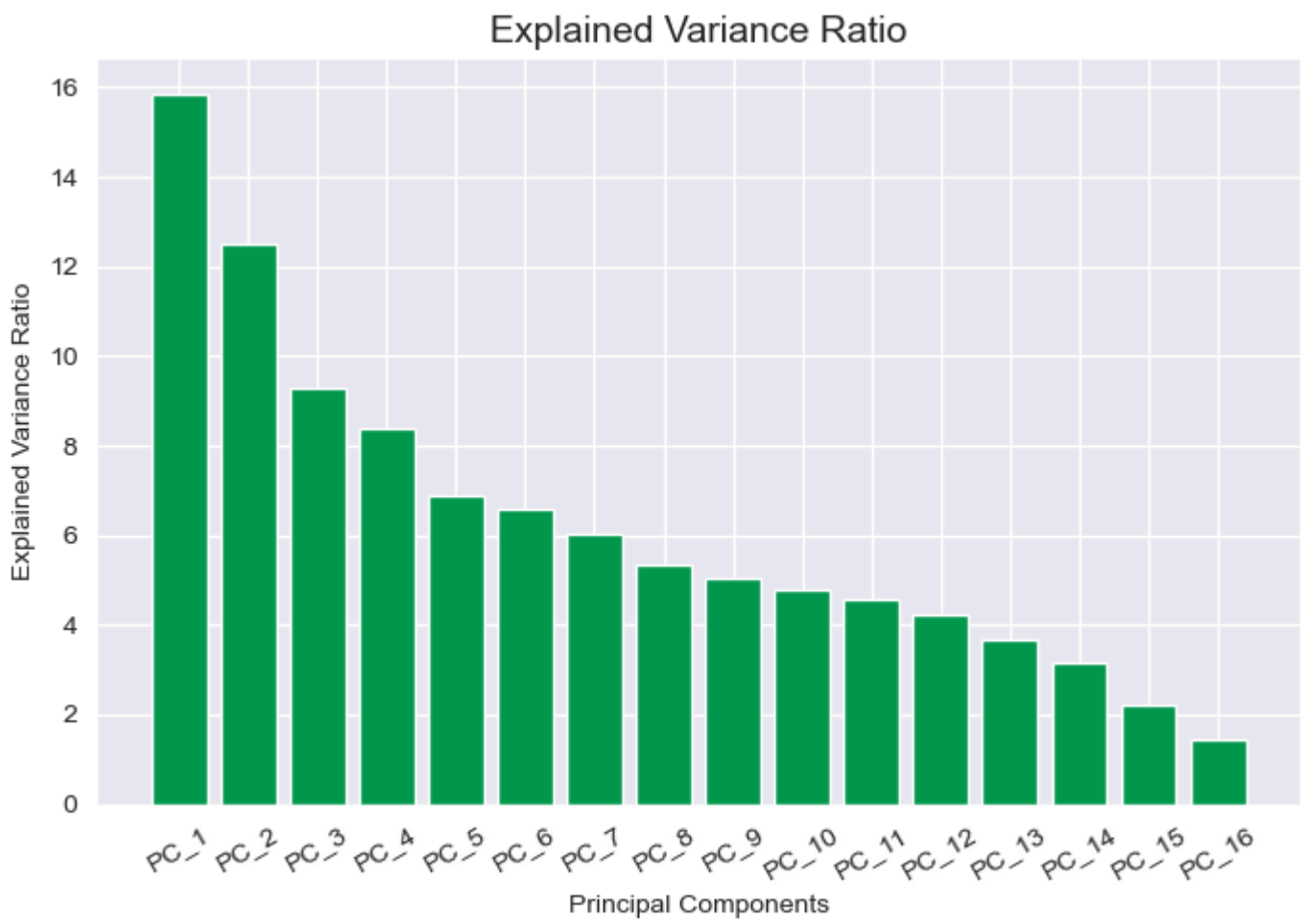
	PC_9	PC_10	PC_11	PC_12	PC_13	PC_14	PC_15	PC_16
Gender	-0.011185	0.009767	-0.163786	0.107444	-0.260021	0.116288	-0.501232	0.424489
Age	0.064068	0.052918	-0.019392	-0.260628	0.042925	0.634342	-0.007275	-0.147871
Height	0.005057	0.111352	-0.053587	0.205917	-0.063163	-0.001829	0.205612	-0.695926
Weight	0.044587	-0.004045	-0.071308	0.203114	0.001570	0.106000	0.586913	0.500846
family_history_with_overweight	0.297016	-0.121875	-0.148702	-0.022611	0.480571	-0.250772	-0.373798	-0.068052
FAVC	0.268523	0.527843	0.499939	-0.278652	-0.158077	-0.073276	-0.075577	0.047124
FCVC	-0.131148	0.322810	0.031376	0.364080	-0.285447	0.167259	-0.380957	-0.046085
NCP	0.096040	-0.591868	0.261295	-0.313136	-0.160775	0.067528	-0.073069	0.046257
CAEC	0.139339	0.415227	-0.482568	-0.236739	0.094855	-0.072565	0.119416	0.138251
SMOKE	-0.296103	0.006337	0.212313	-0.080720	-0.084623	-0.167313	0.012463	0.010603
CH2O	0.039366	0.091064	-0.285232	-0.497222	-0.260641	-0.045014	0.078395	-0.035769
CALC	0.025987	0.020687	-0.193325	0.074209	0.358808	-0.051667	-0.160346	0.016452
SCC	0.724940	-0.001247	0.171767	0.048421	0.019634	0.082477	0.015105	0.004480
FAF	-0.282894	0.224884	0.404110	-0.077728	0.579185	0.178780	0.016511	0.153340
TUE	0.256111	-0.051597	-0.006407	0.379394	0.013756	0.389718	0.075982	0.015297
MTRANS	-0.161257	-0.047863	-0.184392	-0.245931	0.120105	0.493413	-0.116276	-0.084415

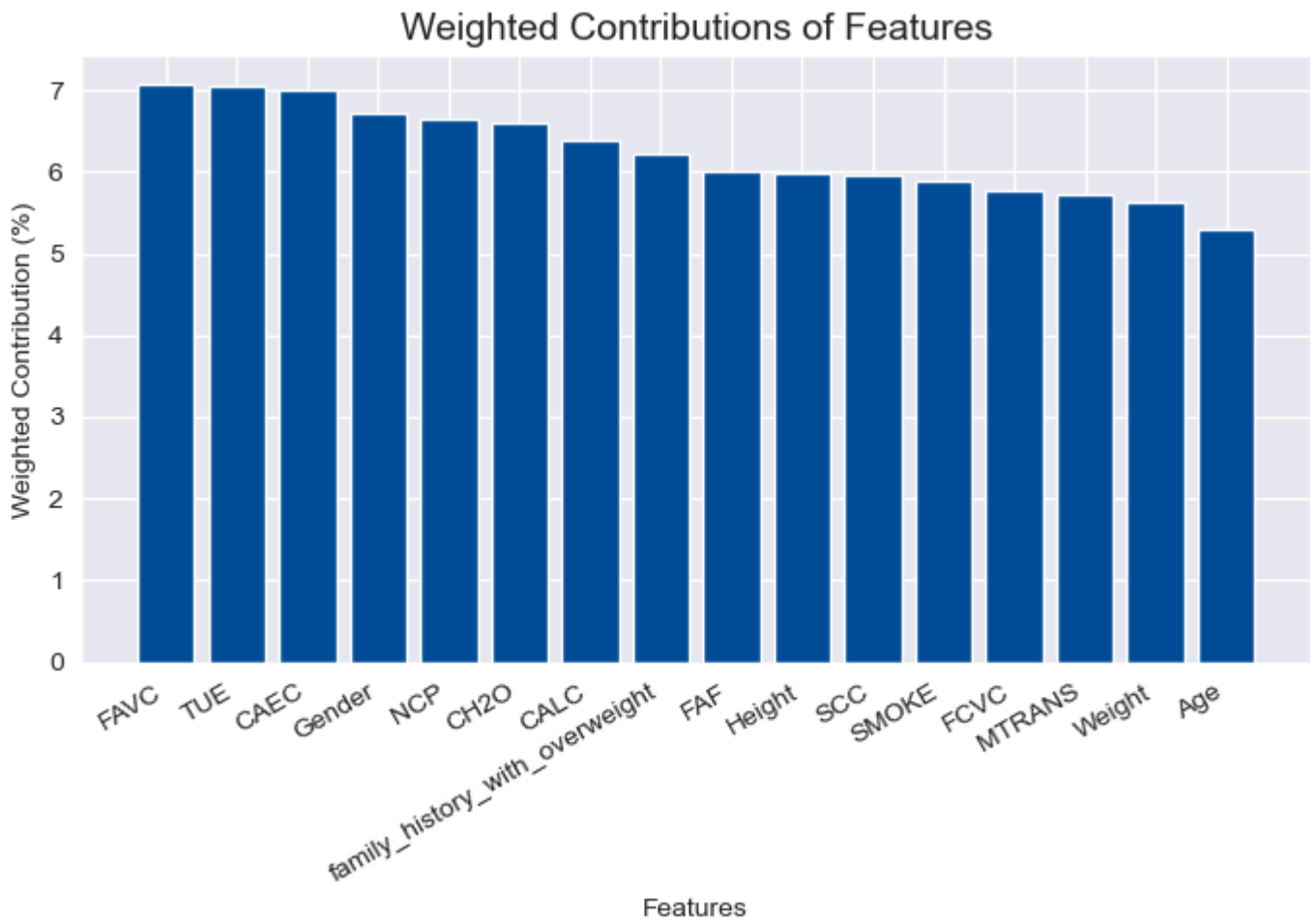
Explained Variance Ratio

	Principal Component	Explained Variance Ratio	Cumulative Variance Ratio
0	PC_1	15.856574	15.856574
1	PC_2	12.496725	28.353300
2	PC_3	9.298448	37.651747
3	PC_4	8.402248	46.053996
4	PC_5	6.890842	52.944838
5	PC_6	6.588940	59.533778
6	PC_7	6.031164	65.564942
7	PC_8	5.352807	70.917749
8	PC_9	5.033045	75.950795
9	PC_10	4.801235	80.752030
10	PC_11	4.578306	85.330336
11	PC_12	4.222632	89.552968
12	PC_13	3.667513	93.220481
13	PC_14	3.134465	96.354946
14	PC_15	2.190911	98.545857
15	PC_16	1.454143	100.000000

Weighted Contributions

	Feature	Weighted Contribution (%)	Cumulative Contribution (%)
0	FAVC	7.070151	7.070151
1	TUE	7.050999	14.121150
2	CAEC	7.009229	21.130379
3	Gender	6.727671	27.858050
4	NCP	6.641395	34.499445
5	CH2O	6.602285	41.101730
6	CALC	6.380768	47.482498
7	family_history_with_overweight	6.226505	53.709003
8	FAF	6.002243	59.711246
9	Height	5.976520	65.687766
10	SCC	5.973553	71.661319
11	SMOKE	5.897297	77.558616
12	FCVC	5.777426	83.336042
13	MTRANS	5.732806	89.068848
14	Weight	5.638483	94.707331
15	Age	5.292669	100.000000



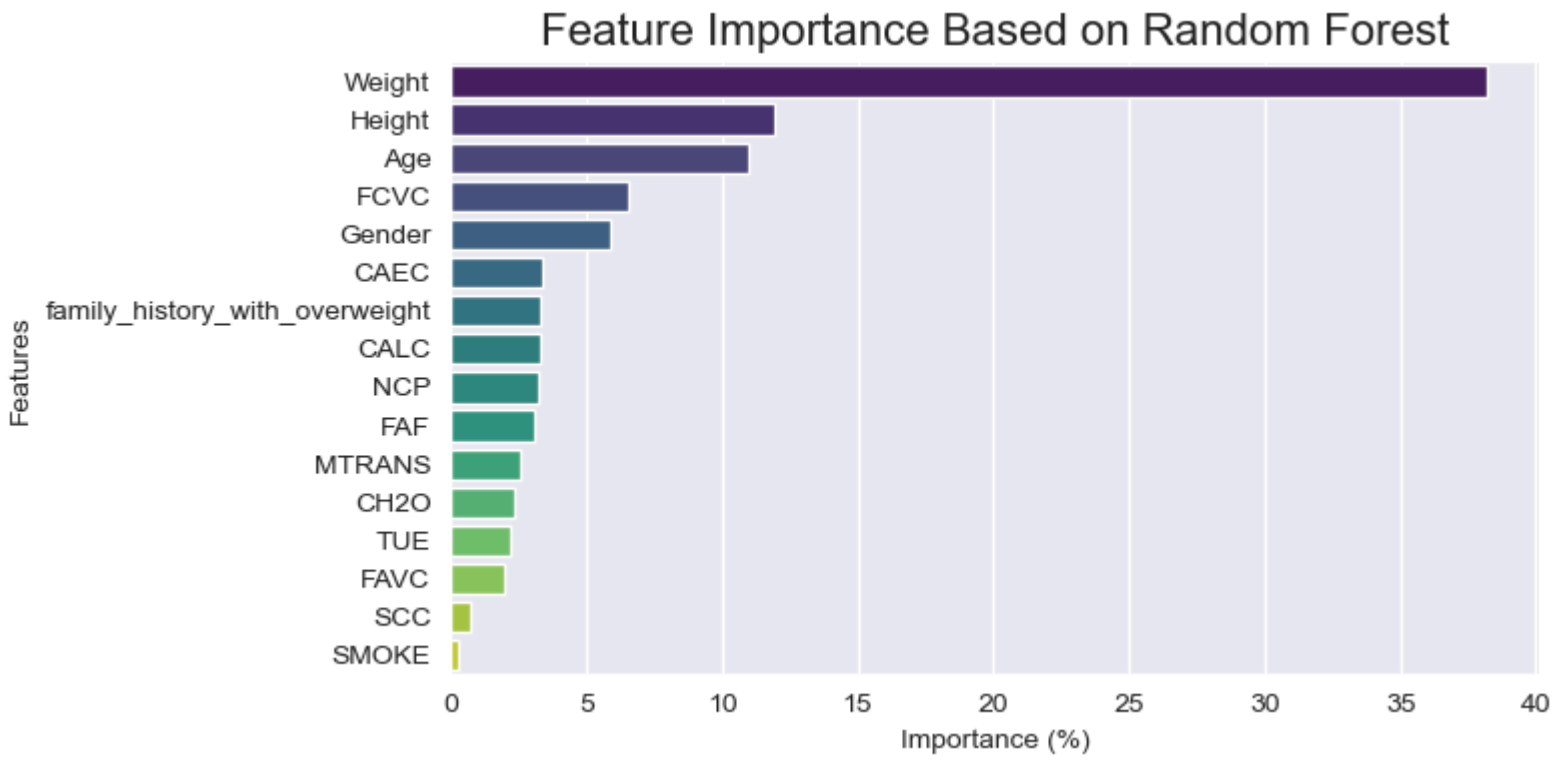



```
  Category  Total Weighted Contribution (%)
0      Eating Habits                45.378550
1  Physical Condition                24.759602
```

In [38]:
```
## Random Forest for variable importance

# calculate importance and visualization
feature_importance = compute_random_forest_importance(df_encoded, basic_features + eating_habits + physical_conditi
display_table_as_html(feature_importance, "Feature Importance Table (Random Forest)")
plot_feature_importance(feature_importance)

# compare importance between Eating Habits and Physical Condition with Random Forest
compare_feature_categories(feature_importance, method='RandomForest')
```

Feature Importance Table (Random Forest)

	Feature	Importance (%)	Cumulative Importance (%)
0	Weight	38.208472	38.208472
1	Height	11.970676	50.179148
2	Age	10.985402	61.164550
3	FCVC	6.589660	67.754210
4	Gender	5.885699	73.639909
5	CAEC	3.361039	77.000949
6	family_history_with_overweight	3.320755	80.321704
7	CALC	3.272868	83.594571
8	NCP	3.264149	86.858720
9	FAF	3.085125	89.943845
10	MTRANS	2.590928	92.534773
11	CH2O	2.344785	94.879558
12	TUE	2.164948	97.044507
13	FAVC	1.945044	98.989550
14	SCC	0.756335	99.745885
15	SMOKE	0.254115	100.000000




```
            Category  Total Weighted Contribution (%)
0       Eating Habits                        21.031658
1  Physical Condition                         8.597337
```

Objective 4: Traditional Model Building

Explanation of Results

Basically, the dataset is suitable to employ non-linear regression to do estimation tasks. So, Logistical Regression and SVM methods are selected to simulate the estimation of the data after completing some comparison trials on several mainstream non-linear regression methods. The explanation of the selection of such models can be referred to in [2], [3].

Outcomes of Methodology 1: Modeling after PCA

After test all the potential numbers of the selected features, both Logistic Regression and SVM had better to `join all the 16 features` into the modeling so that the performance can be best in all scales of evaluation indexes.

The best performance (Avg-ACC) is:

- 75% (`Logistical Regression`)
- 71% (`SVM`)

Outcomes of Methodology 2: Modeling after Random Forest

After test all the potential numbers of the selected features, both Logistic Regression and SVM had better to `join the first 14 features` into the modeling so that the performance can be best in all scales of evaluation indexes.

The best performance (Avg-ACC) is:

- 75% (`Logistical Regression`)
- 71% (`SVM`)

Conclusion on the outcomes from two methods

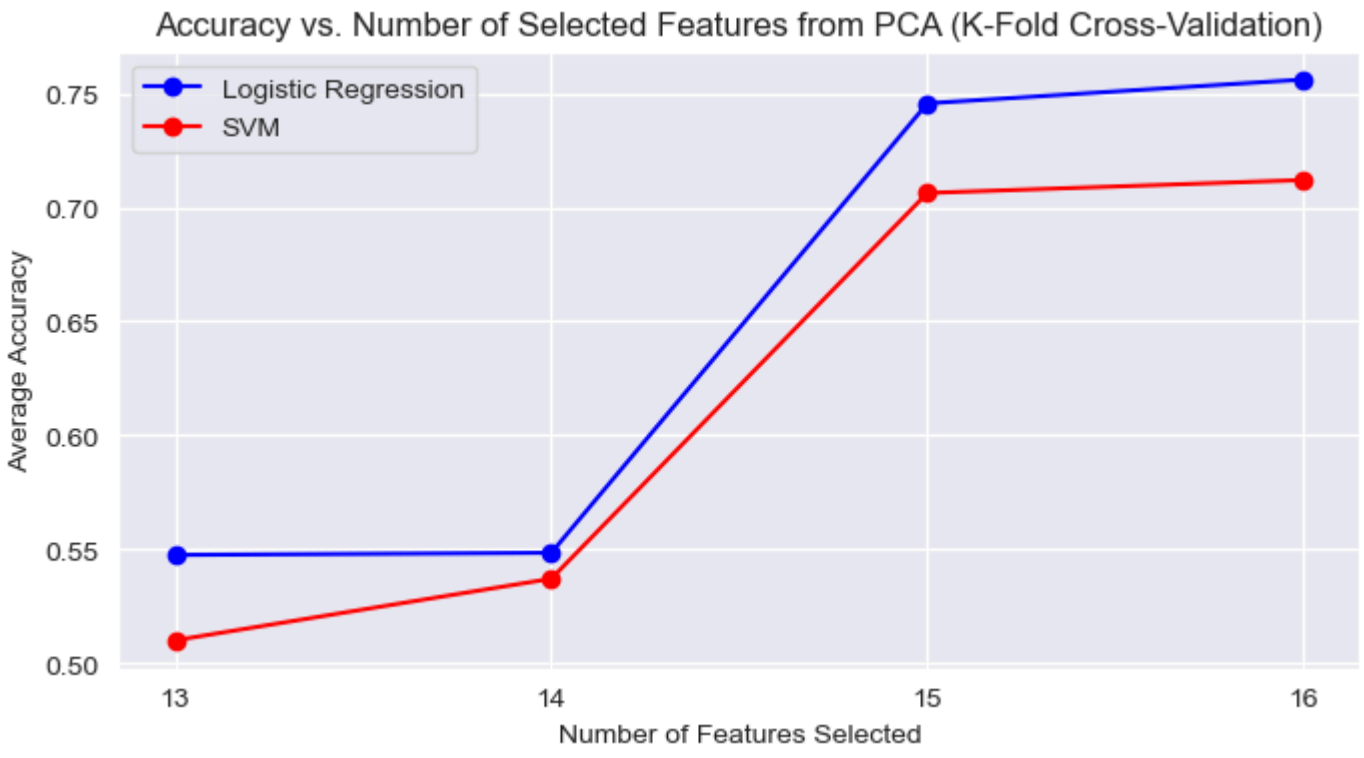
From the results above, we can make the conclusions:

1. The `overall performance` of the models, based on the two important feature selection methods, `show little differentiation for the certain evaluation index` .
2. The `important feature selection schemes with Random Forest performs relatively better` because the final mode shows the `similar accuracy` , but the model directed by Random Forest `utilize fewer features when modeling` . This is the purpose of the dimension reduction, and in this way we could simplify the model without influencing the performance of it.

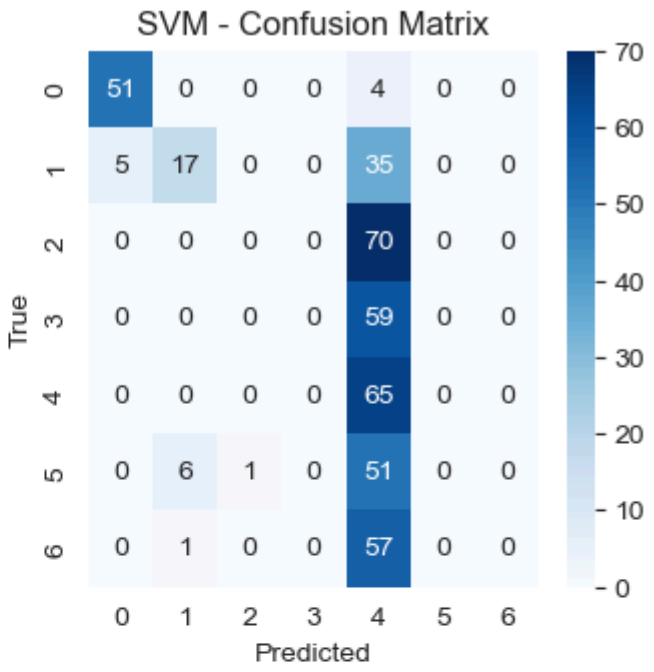
Visualisation

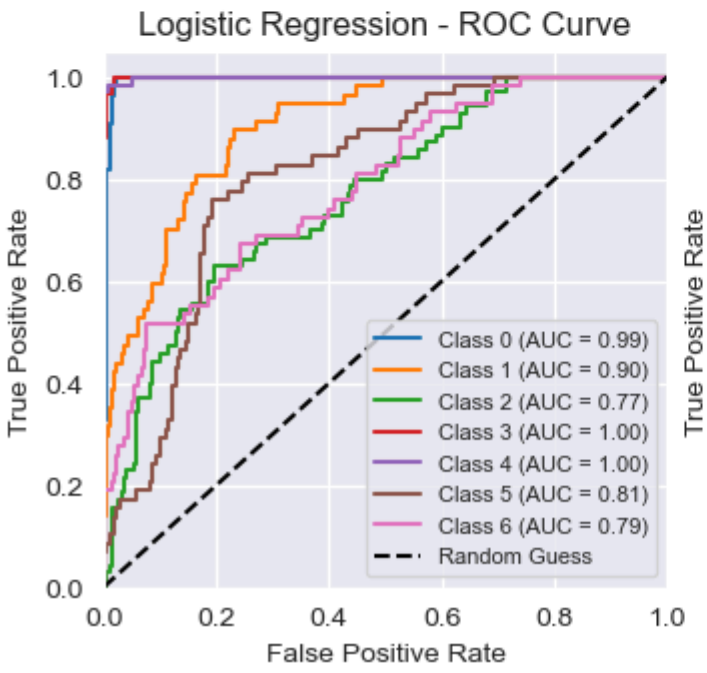
Below are the visualization of the Logistical Regression and SVM modeling for estimation of obesity level, making `comparisons about the performance between the variable importance results from PCA and Random Forest` . Also, the visualization of the `evaluation indexes` are properly printed, together with `confusion matrix` and `ROC Curve` .

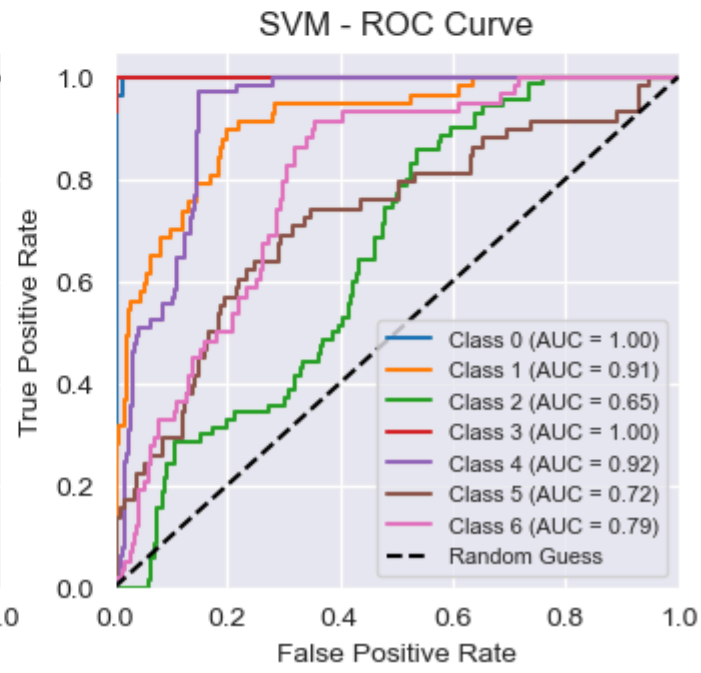
In [49]:
```
# build and evaluate model with selected features by PCA
evaluate_and_plot_feature_selection(method='PCA')
```



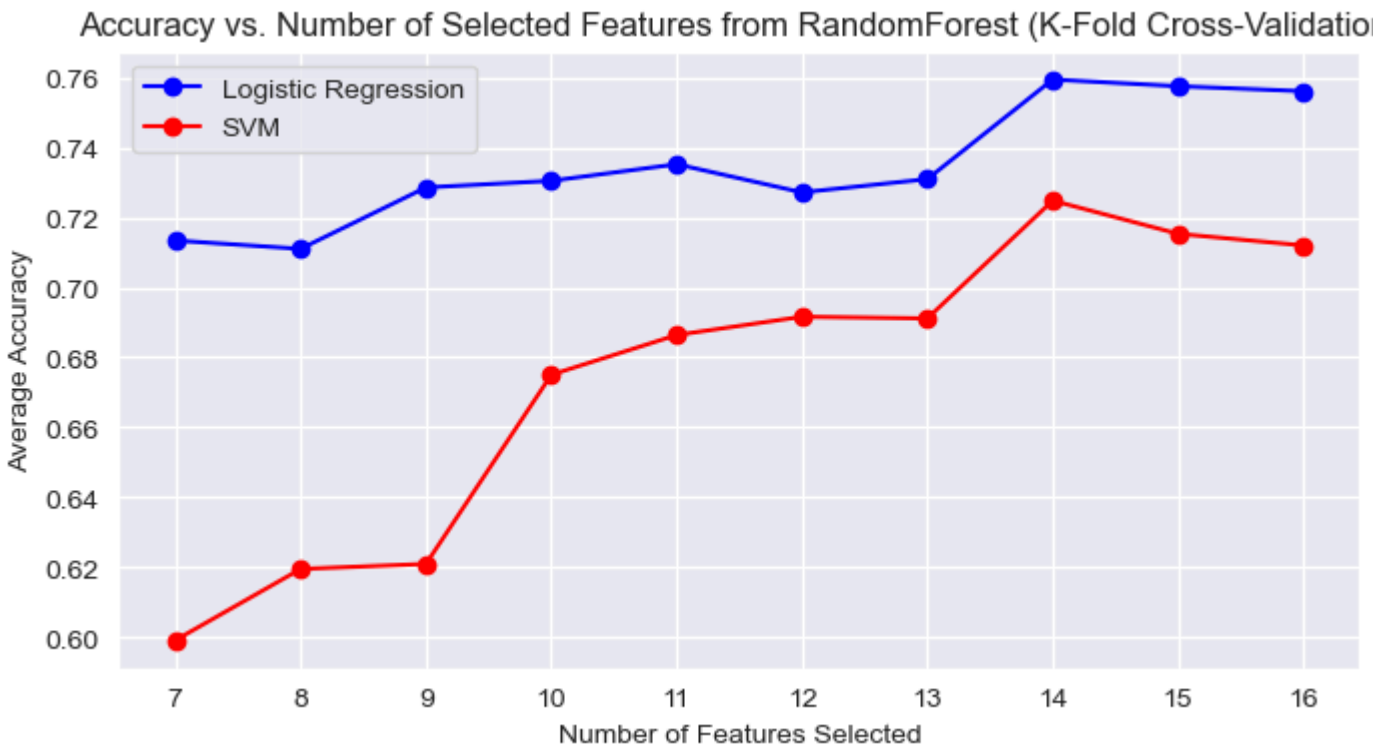

```
===== Best Logistic Regression Model Evaluation Based on PCA (Using Top 16 Features) =====

Logistic Regression Model Performance:
              precision    recall  f1-score   support

           0       0.84      0.98      0.91        55
           1       0.76      0.56      0.65        57
           2       0.64      0.60      0.62        70
           3       0.77      1.00      0.87        59
           4       0.96      1.00      0.98        65
           5       0.60      0.66      0.63        58
           6       0.67      0.48      0.56        58

    accuracy                           0.75       422
   macro avg       0.75      0.75      0.74       422
weighted avg       0.75      0.75      0.74       422


===== Best SVM Model Evaluation Based on PCA (Using Top 16 Features) =====

SVM Model Performance:
              precision    recall  f1-score   support

           0       0.87      0.98      0.92        55
           1       0.61      0.54      0.57        57
           2       0.59      0.51      0.55        70
           3       0.79      1.00      0.88        59
           4       0.98      1.00      0.99        65
           5       0.52      0.55      0.54        58
           6       0.50      0.40      0.44        58

    accuracy                           0.71       422
   macro avg       0.70      0.71      0.70       422
weighted avg       0.70      0.71      0.70       422
```



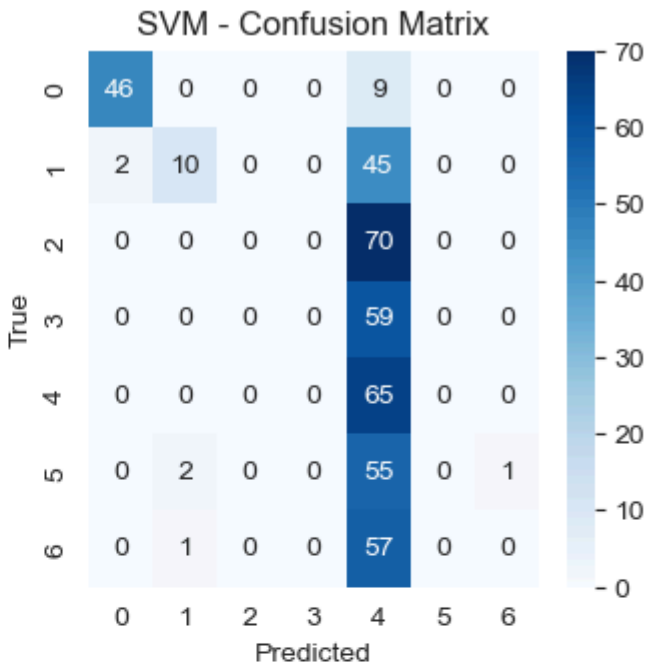


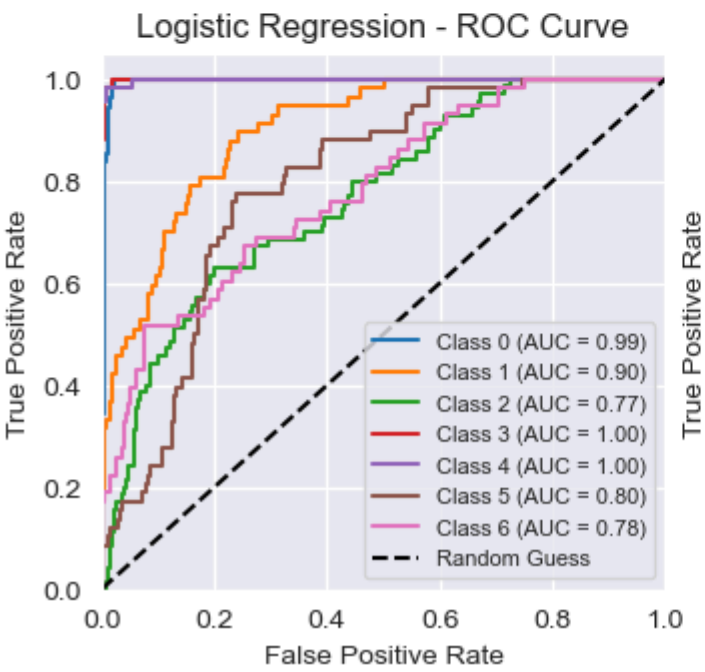


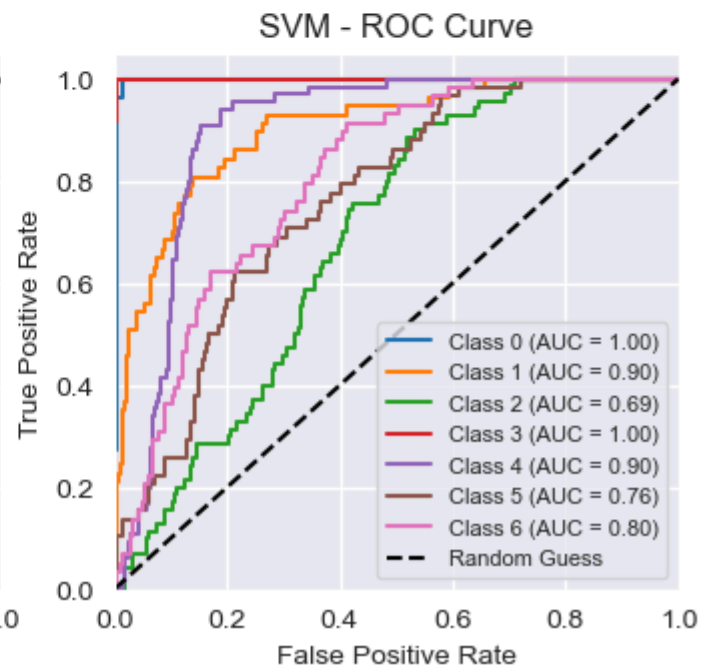
In [50]:
```
# build and evaluate model with selected features by Random Forest
evaluate_and_plot_feature_selection(method='RandomForest')
```



```
===== Best Logistic Regression Model Evaluation Based on RandomForest (Using Top 14 Features) =====

Logistic Regression Model Performance:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        55
           1       0.73      0.58      0.65        57
           2       0.63      0.60      0.61        70
           3       0.77      1.00      0.87        59
           4       0.98      1.00      0.99        65
           5       0.60      0.60      0.60        58
           6       0.61      0.47      0.53        58

    accuracy                           0.75       422
   macro avg       0.74      0.75      0.74       422
weighted avg       0.74      0.75      0.74       422


===== Best SVM Model Evaluation Based on RandomForest (Using Top 14 Features) =====

SVM Model Performance:
              precision    recall  f1-score   support

           0       0.86      1.00      0.92        55
           1       0.64      0.60      0.62        57
           2       0.58      0.36      0.44        70
           3       0.78      1.00      0.87        59
           4       0.98      1.00      0.99        65
           5       0.46      0.59      0.52        58
           6       0.57      0.45      0.50        58

    accuracy                           0.71       422
   macro avg       0.70      0.71      0.70       422
weighted avg       0.70      0.71      0.69       422
```







Objective 5: Advanced Modeling

Explanation of Results

For advanced modeling, the dataset was evaluated using more complex models like `XGBoost`, `TabNet`, and a hybrid combination of `XGB + TabNet`. These models are well-suited for capturing non-linear interactions and complex relationships in the table data and discrete features, making them ideal for the estimation tasks presented by the dataset. The selection of these models can be justified based on the capability of ensemble and deep learning-based models in dealing with complex feature sets, as is shown in [4], [5].

Hyperparameter Settings

In regard to the hyperparameters in ML and DL models, they can be concluded as follows:

1. **`XGBoost`**

`Basic settings`

- n_estimators=50
- eval_metric='mlogloss'
- random_state=9
- verbosity=0

`Training process related settings`

- iteration_range=(0, epoch + 1)
- eval_set

2. **`TabNet`**

`Basic settings`

- device_name='cpu'
- verbose=0

`Training process related settings`

- max_epochs=21
- patience=50
- batch_size=1024
- virtual_batch_size=128

3. **`XGBoost + TabNet`**

`Settings`

- XGBoost part: Same as the separate XGBoost setting, used as a feature converter to convert input data into a feature representation (the 'apply' method).
- TabNet part: Same as the separate TabNet setup, but using the features transformed by XGBoost as input.

Detailed results

After testing various numbers of selected features, the advanced models (XGBoost, TabNet, XGB + TabNet) demonstrated the best performance when `all 16 features` were included in the modeling. This indicates that the models benefit from the complete feature set, leveraging the rich information available in each feature.

The best performance (Avg-ACC) is:

- 96% (`XGBoost`)
- 92% (`TabNet`)
- 82% (`XGB + TabNet`)

Conclusion on the Outcomes

Based on the results, the following conclusions can be drawn:

1. The `XGBoost` algorithm performs significantly better than any other ones with `the accuracy of 96%` , followed by sole `TabNet` , `XGB + TabNet` , `92% and 82%` respectively.
2. It can be concluded that the `machine learning model seems most suitable for this dataset` with certain properties (not a large scale dataset, and data imbalance exists), and the TabNet is also a reliable method to deal with table dataset (just correspond to the attributes of the TabNet in [5]).
3. The combination of these methods performs worst, which means the `single algorithm is reliable and enough to do the multi-class classification task in this dataset` , and `this type of model combination is anything but beneficial to the performance` .

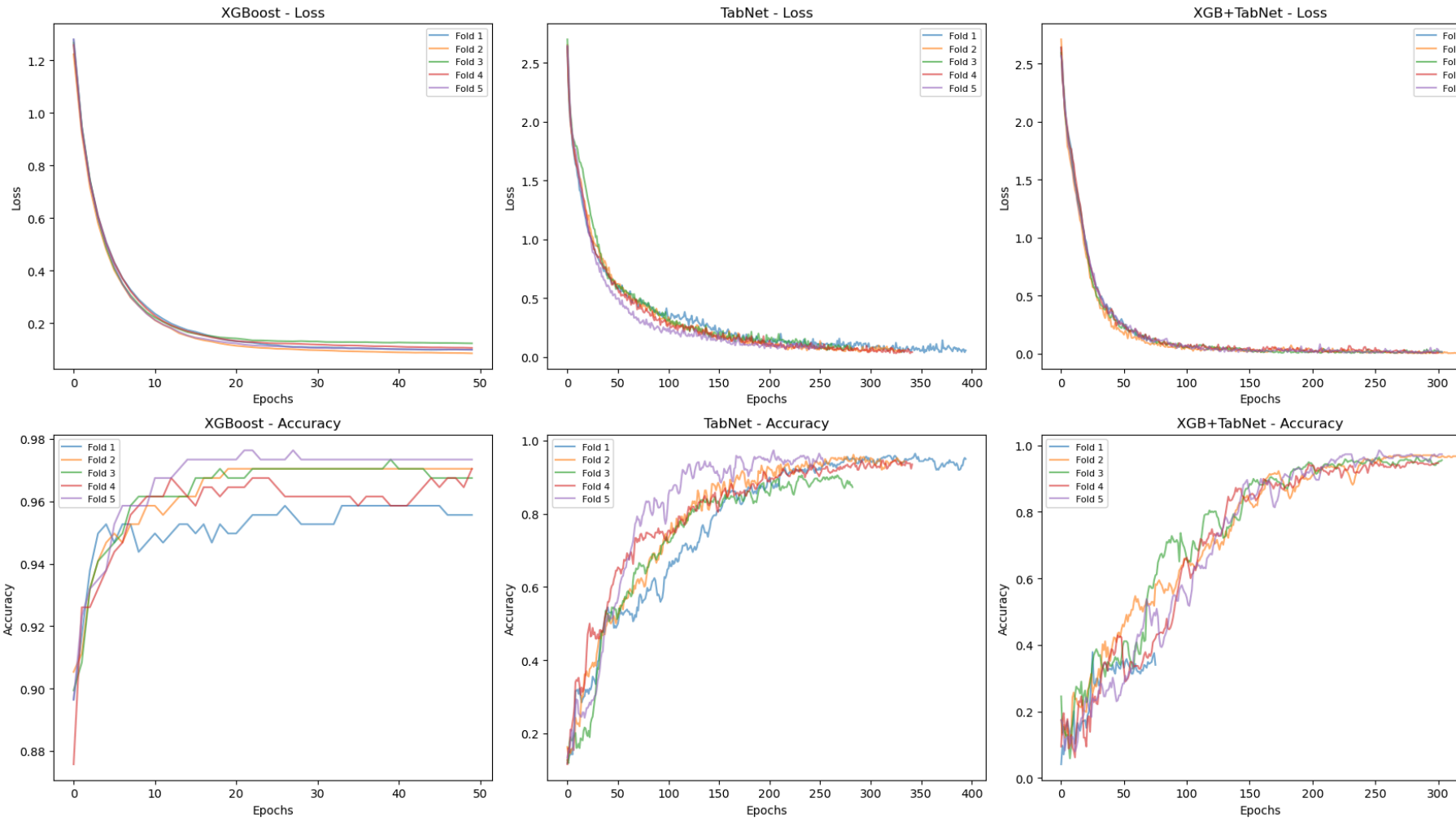
Visualisation

Below are the visualizations for XGBoost, TabNet, and XGB + TabNet models for the estimation of obesity level, including the `precess of iteration` (early stopping is set here) and the `evaluation outputs` of the model. Besides, the process of iteration for the `Accuracy` and `Loss` is visualized in the plots.

In [23]:
```
## advanced modeling
feature_columns = basic_features + eating_habits + physical_condition
X = df_encoded[feature_columns]
y = df_encoded['ObesityLevel']
```

```
# evaluation and visualization using K-fold Cross-Va
evaluate_models_kfold(models, X, y, n_splits=5)
```

```
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 394 with best_epoch = 344 and best_val_0_accuracy = 0.9645
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 75 with best_epoch = 25 and best_val_0_accuracy = 0.3787
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 333 with best_epoch = 283 and best_val_0_accuracy = 0.96154
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 317 with best_epoch = 267 and best_val_0_accuracy = 0.97041
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 282 with best_epoch = 232 and best_val_0_accuracy = 0.90533
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 303 with best_epoch = 253 and best_val_0_accuracy = 0.9645
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 341 with best_epoch = 291 and best_val_0_accuracy = 0.94675
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 300 with best_epoch = 250 and best_val_0_accuracy = 0.9497
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 254 with best_epoch = 204 and best_val_0_accuracy = 0.97337
Training:   0%|          | 0/500 [00:00<?, ?epoch/s]
Early stopping occurred at epoch 303 with best_epoch = 253 and best_val_0_accuracy = 0.98521
XGBoost Model Average Accuracy: 0.96
Classification Report for XGBoost Model (Averaged over 5 folds):
              f1-score  precision    recall   support
0             0.965724   0.958115  0.974696   54.4000
1             0.913409   0.923823  0.904834   57.4000
2             0.974107   0.970974  0.977905   70.2000
3             0.987856   0.993355  0.982558   59.4000
4             0.996882   1.000000  0.993812   64.8000
5             0.918539   0.911785  0.927082   58.0000
6             0.965944   0.968395  0.964314   58.0000
accuracy      0.962100   0.962100  0.962100    0.9621
macro avg     0.960351   0.960921  0.960743  422.2000
weighted avg  0.962086   0.962956  0.962100  422.2000
TabNet Model Average Accuracy: 0.92
Classification Report for TabNet Model (Averaged over 5 folds):
              f1-score  precision    recall     support
0             0.944503   0.948249  0.945525   54.400000
1             0.851085   0.860604  0.846138   57.400000
2             0.939255   0.940692  0.938825   70.200000
3             0.965290   0.979061  0.952405   59.400000
4             0.995378   1.000000  0.990827   64.800000
5             0.856100   0.841824  0.873035   58.000000
6             0.902595   0.891566  0.915484   58.000000
accuracy      0.924667   0.924667  0.924667    0.924667
macro avg     0.922029   0.923142  0.923177  422.200000
weighted avg  0.925009   0.927480  0.924667  422.200000
XGB+TabNet Model Average Accuracy: 0.82
Classification Report for XGB+TabNet Model (Averaged over 5 folds):
              f1-score  precision    recall     support
0             0.857100   0.945458  0.829570   54.400000
1             0.761322   0.776768  0.750556   57.400000
2             0.816298   0.800511  0.837988   70.200000
3             0.891409   0.883513  0.914341   59.400000
4             0.892800   0.880287  0.907708   64.800000
5             0.717150   0.727260  0.722323   58.000000
6             0.767195   0.769550  0.765483   58.000000
accuracy      0.820687   0.820687  0.820687    0.820687
macro avg     0.814753   0.826193  0.818281  422.200000
weighted avg  0.816393   0.825372  0.820687  422.200000
```



Conclusion (5 marks)

Achievements

The main achievements can be concluded as follows.

1. The visualization of the distribution patterns found the data imbalance of the certain features.
2. The correlation analysis explain the non-linear relation between the features and the target, and the nonexistence of multi-collinearity.
3. The variable importance analysis by PCA and Random Forest found the contribution of each features to the target. Besides, relatively, the eating habits matters more than physical condition when it comes to obesity alleviation.
4. The traditional regression modeling suggested that the Logistical Regression and SVM are suitable for the non-linear tasks. The result of variable importance analysis by Random Forest seemed more reliable compared with PCA, providing the experience for modeling based on such kind of dataset.
5. The advanced ML and DL modeling gave some insights that if the dataset is quite small, the ML methods are still a wise and reliable choice, while the DL models usually require a large quantity of data to perform better. In addition, certain early stopping consideration is supposed to be included when it comes to DL, because this module has the advantage to saving computing resources and prevent meaningless iteration.

Limitations

1. `Data imbalance`:
   The dataset was collected by surveys and simulated and generated by certain approaches to expand the scale of it. However, although the imbalance of obesity level was correctly rectified, the imbalance in several features still exists. It may be partly because the form of the data collection was online, and the elderly people would show up less, leading to the data imbalance in 'Age' and less information about this age group being extracted from it.
2. `Synthetic data and lack of original records`:
   In the 2111 records available, 77% of them were generated using techniques such as Weka tools and SMOTE filter to balance the obesity levels. This synthetic augmentation points to a lack of sufficient original record which inherently limits the validity of any conclusions drawn from the data. When synthetic data forms a majority of the dataset, it can also distort the underlying relationships and make it challenging to apply certain statistical and machine learning methods effectively, potentially reducing the generalization ability of the results to real-world scenarios.

Future Work

1. `Collection of the larger scale of original data`:
   For future work, it would be beneficial to collect more diverse and extensive original data, including more variables that influence obesity. By expanding the dataset to include more original, non-synthetic records, we aim to ensure a more robust foundation for analysis. This would make impossible to explore different relationships more accurately. while avoiding over-reliance on generated data.
2. `Expansion of features to give information from more various dimensions`:
   In regard to the 16 features, it seems that they are just enough to participate in the traditional regression modeling. However, if more

features are considered and added into the estimation task, the measurement of the important factors and the performance of the model could become more satisfied since the accuracy of the traditional model is not pleasant enough.

3. `Development of a new overall index for obesity level quantification`:
   The index could be a more complex relationship between height, weight, and other factors that impact obesity, while the Body Mass Index (BMI) approach is employed to quantify the obesity level. Therefore, developing a new index to assess body mass in relation to different eating habits physical activity, and individual health profiles could yield more nuanced insights. These extended variables could help better characterize obesity, enabling more and personalized accurate analysis.

References

[1] Estimation of Obesity Levels Based On Eating Habits and Physical Condition [Dataset]. (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5H31Z. [2] Huang, Shuai. "Obesity prediction based on logistic regression, random forest and support vector machine." (2022). [3] Wong, Jyh Eiin, et al. "Predicting overweight and obesity status among Malaysian working adults with machine learning or logistic regression: retrospective comparison study." JMIR Formative Research 6.12 (2022): e40404. [4] Prasher, Shikha, and Leema Nelson. "Early Prediction of Obesity Risk in Older Adults using XGBoost Classifier." 2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT). Vol. 1. IEEE, 2024. [5] Arik, Sercan Ö., and Tomas Pfister. "Tabnet: Attentive interpretable tabular learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 35. No. 8. 2021.