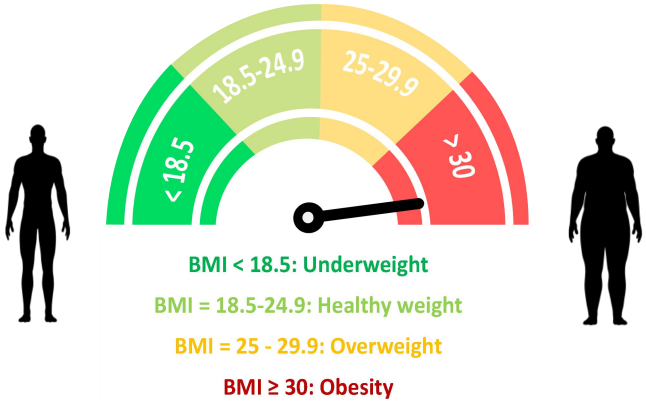
ISyE 6740 – Spring 2024
Project Report



TEAM MEMBER NAMES: Batoulsadat Haerian (Term Project 083)

PROJECT TITLE: Machine Learning-Based Analysis of Obesity Risk Factors

INTRODUCTION

Problem Statement

Excessive fat accumulation, characterized by overweight and obesity, is a global public health concern. Individuals with a body mass index (BMI) exceeding 25 are classified as overweight, while those with a BMI surpassing 30 are categorized as obese. Alarmingly, in 2019 alone, an estimated 5 million deaths attributed to noncommunicable diseases (NCDs) were linked to BMI levels higher than the optimal range.[1] Globally, the prevalence of obesity from 38% in 2020 will increase to 51% (approximately 4 billion individuals aged over 5 years) by 2035 unless significant improvements are made in prevention, treatment, and support strategies. Particularly concerning is the rapid escalation of obesity rates among children and in lower-income countries. The implications of obesity extend from individual health to healthcare systems and economies worldwide.[2]

Obesity increases risk of cardiovascular diseases, hypertension, stroke, dyslipidemia, type 2 diabetes, osteoarthritis, and certain cancers. Therefore, it is a substantial burden on healthcare resources and budgets.[3] The economic costs of obesity and its associated health consequences is projected to increase from 1.96 trillion to over US$4 trillion by 2035. This is a significant strain on healthcare systems, insurance providers, employers, and government budgets, impacting economic productivity and overall societal well-being. Thus, addressing the obesity epidemic is paramount not only for safeguarding public health but also for alleviating the economic burdens and enhancing the quality of life for individuals and communities.[4]

Obesity has a complex nature due to the contribution of genes and external factors such as diet, physical activity levels, socioeconomic status, and cultural influences converge to shape individual susceptibility to obesity.[5] Understanding the effect of the primary risk elements is crucial for formulating robust prevention, intervention strategies, and management of obesity.

Despite a wide research on obesity and its associated diseases and interventions targeting nutrient intake, diet, and physical activity, yet, their efficiency remains crucial. Machine learning (ML) algorithms are valuable tools for biomarker detection and intervention strategy discovery. These techniques provide a robust framework for analyzing intricate datasets and uncovering patterns that may influence obesity risk.[6] However, the extent to which these algorithms can accurately predict obesity risk factors based on demographic and lifestyle variables remains unclear. Additionally, determining the optimal model for predicting obesity risk using the specified predictors and evaluating the impact of various data pre-processing techniques on predictive accuracy are essential considerations. Furthermore, understanding how different evaluation metrics aid in assessing the

performance of predictive models is crucial. This research aims to leverage advanced ML methodologies to conduct a comprehensive analysis of obesity risk factors, encompassing demographic and lifestyle elements. By addressing these questions, the study seeks to enhance insights into the multifaceted nature of obesity and underscore the potential of ML in advancing obesity research and interventions.

Hypothesis

1. ML algorithm classifications accurately predict obesity risk factors using demographic and lifestyle variables.
2. The predictive performance of obesity risk factors varies across different machine learning algorithm classifications.

METHODOLOGY

1. Data Source

This project will employ ML techniques on the synthetic dataset "Multi-Class Prediction of Obesity Risk," featured in the 2024 Kaggle Playground Series competition. The database consisted of a zip file containing two datasets: train (20,758 rows, 18 columns) and test (13,840 rows, 17 columns), with features closely resembling those of the original dataset. Generated through a deep learning model, each entry includes 18 attributes (Table 1).[7]

Table 1. The first five rows of the synthetic train dataset (20,758 rows, 18 columns).

id	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Male	24.44301	1.699998	81.66995	yes	yes	2	2.983297	Sometimes	no	2.763573	no	0	0.976473	Sometimes	Public_Transportation	Overweight_Level_II
1	Female	18	1.56	57	yes	yes	2	3	Frequently	no	2	no	1	1	no	Automobile	Normal_Weight
2	Female	18	1.71146	50.165754	yes	yes	1.880534	1.411685	Sometimes	no	1.910378	no	0.866045	1.673584	no	Public_Transportation	Insufficient_Weight
3	Female	20.95274	1.71073	131.274851	yes	yes	3	3	Sometimes	no	1.674061	no	1.467863	0.780199	Sometimes	Public_Transportation	Obesity_Type_III
4	Male	31.64108	1.914186	93.798055	yes	yes	2.679664	1.971472	Sometimes	no	1.979848	no	1.967973	0.931721	Sometimes	Public_Transportation	Overweight_Level_II
5	Male	18.12825	1.748524	51.552595	yes	yes	2.919751	3	Sometimes	no	2.13755	no	1.930033	1	Sometimes	Public_Transportation	Insufficient_Weight

Originally compiled in 2019, the dataset consisted of 2,111 records. Approximately 77% of the data was synthesized from 485 records using the Weka tool and the SMOTE filter. Data collection involved an online survey accessible for 30 days, with participants from Mexico, Peru, and Colombia providing estimates of their obesity levels based on dietary patterns and physical health conditions.[8] For a detailed overview of the dataset, refer to Table 2.

Table 2. An overview of the table parameters.

Parameters	Description	Values
1. Demographic (predictive) variables		
id	-	-
Gender	Gender	Female, Male
Age	Age	Year
Height	Height	Meter
Weight	Weight	kilogram
family_history_with_overweight	Family history with overweight	Yes , No
2. Lifestyle (predictive) variables		
FAVC	Frequently eating high caloric food	Yes, No
FCVC	Eating vegetables in meals	Never, Sometimes, Always
NCP	Number of daily main meals	1-2, 3, >3
CAEC	Eating any food between meals	No, Sometimes, Frequently, Always
SMOKE	Smoking	Yes, No
CH2O	Daily drinking water	< 1 liter, 1-2 L, > 2 L
SCC	Monitoring daily eating calories	Yes, No
FAF	Frequency of weekly physical activity	Never, 1 or 2 days, 2 or 4 days, 4 or 5 days
TUE	Number of hours using technological devices (cell phone, video-games, television, computer, etc.)	0–2 hours, 3–5 hours, > 5 hours
CALC	Frequency of drinking alcohol	Never, Sometimes, Frequently, Always
MTRANS	Transportation type	Automobile, Motorbike, Bike, Public Transportation, Walking
3. Response variable		
NObeyesdad	Body mass Index (BMI) = Weight/Height^2	Underweight: < 18.5, Normal: 18.5-24.9 Overweight: 25.0-29.9, Obesity I: 30.0-34.9 Obesity II: 35.0-39.9, Obesity III: ≥ 40

After downloading the train and test datasets in CSV format from Kaggle, they were consolidated into a unified table to adjust the split ratio from 0.6 to 0.75 for further pre-processing and analysis.

2. Data Preprocessing

Data Cleaning: The BMI was calculated for each individual using the formula weight/height^2, and the resulting BMI values were stored in a new column named "bmi". Subsequently, the "NObeyesdad" column was replaced with the "bmi" column. Individuals were then categorized into BMI groups based on established thresholds: Underweight (< 18.5), Normal (18.5-24.9), Overweight (25.0-29.9), and Obesity (≥ 30.0), and these categories were recorded in the "bmi_group" column.[9] The "bmi_binary" column was also generated. After excluding the underweight category (BMI < 18.5) from the BMI_binary column, the value of 0 was assigned if the BMI fell within the normal range (18.5-24.9), while a value of 1 was assigned if it exceeded 25, indicating overweight or obesity. As a result, the dimensions of the final dataset became 30,537 rows and 20 columns.

Handling Missing Data: Missing data was only identified in the "NObeyesdad" column of the train dataset. Upon replacing this column with the "bmi" column, no further missing data were observed in the merged table.

Encoding Categorical Variables: Given that the Kaggle synthetic dataset represented categorical variables FCVC, NCP, CH2O, FAF, and TUE as float values, they were converted to categorical type for consistency and appropriate analysis.

Dealing the outliers: Two approaches were considered for handling outliers. Firstly, outliers were identified using z-scores and trimmed if they exceeded 3 standard deviations from the mean. Secondly, the dataset was split into non-outliers (28,679 rows and 14 columns) and outliers (1858 rows and 14 columns) for further analysis to determine the impact of outliers on the results.

Multicollinearity: The correlation between predictors will be assessed using correlation matrices, VIF factors, and visualized through heatmaps. Predictors with a correlation coefficient $r > 0.4$ and a VIF > 4 will be considered indicative of high multicollinearity and may be excluded from the analysis.[10]

Feature Selection: Key predictors include 'Age', 'age_capped', 'age_group', 'Gender', 'FH', 'FAVC', 'FCVC', 'NCP', 'CAEC', 'Smoke', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC', and 'Transportation' with the key dependent variable being 'bmi_binary.'

Data Transformation: Two approaches were considered for allocating the dataset into training and test datasets. The first approach involved random splitting, where approximately 75% of entries were allocated to the train set (22,902 rows and 21 columns), and the remaining 25% formed the test set (7,635 rows and 21 columns) for further analysis. The second approach utilized 10-fold cross-validation for random allocation of train and test sets, followed by prediction analysis.

3. ML Classifiers for Predictive Modeling

Predictive Models: To ensure more reliable classification results, we employed several algorithms for predicting obesity risk factors: Logistic Regression (LoR), Decision Trees (DTs), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Neural Networks (NNs). In these models, the target variable is binary, representing whether an individual is at risk of obesity. Specifically, we considered the column "BMI_binary" as the dependent variable, where a value of 1 indicates a BMI between 18.5 and 24.9, and a value of 2 indicates a BMI greater than or equal to 25.

LoR estimates the probability of individuals being at risk of obesity based on predictor variables, offering insights into contributing factors. However, LoR is sensitive to multicollinearity, leading to unstable coefficient estimates and inflated standard errors.[11]

DTs predict obesity risk factors hierarchically, classifying individuals into risk levels based on specific criteria. While informative, DTs are sensitive to outliers, which can distort the tree structure and decision boundaries, causing overfitting.[12-13]

RF employs multiple DTs to enhance accuracy and identify influential obesity risk factors. Resilient to outliers, RF is susceptible to redundant or irrelevant features.[13-14]

SVM categorizes individuals into risk groups using a hyperplane that maximizes class margin. Its performance depends on feature scaling, as SVM relies on distances between data points to establish decision boundaries.[12-13]

KNN predicts obesity risk factors based on similarity to nearest neighbors. Its effectiveness relies on distance metric choice and feature scaling.[12]

NNs learn complex patterns for precise predictions, capturing nonlinear relationships in obesity risk. However, NNs are sensitive to initial weight and bias setup, affecting model convergence and performance.[13,15]

For evaluating model performance, Accuracy, Precision, Recall, and F1 Score metrics will be used. These metrics will be compared across classifiers, considering datasets with and without age outliers, utilizing both test-train split and 10-fold cross-validation (CV) approaches for each model.

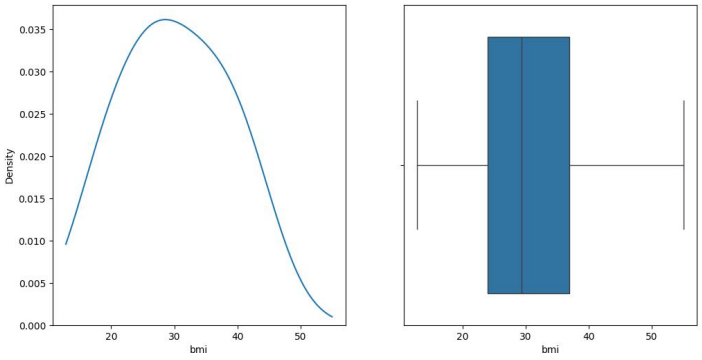
Table 3. An overview of the metrics used evaluation of model performance.

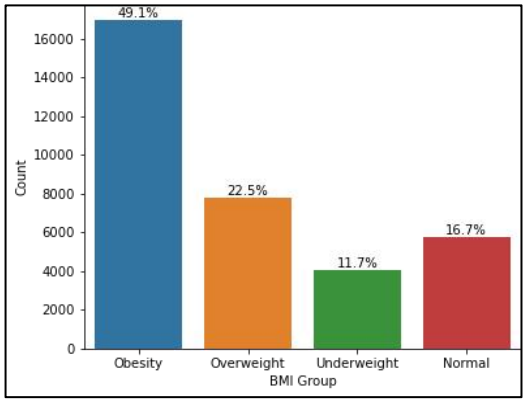
Metrics	Description
Accuracy	the proportion of correctly classified instances among all instances in the dataset, providing an overall evaluation of classification model performance across different classes.
Precision	It serves as a positive predictive value, representing the ratio of true positive (TP) predictions to the sum of TP and false positive (FP) predictions.
Recall	It denotes the model's sensitivity in detecting positive outcomes, computed as the ratio of TP predictions to the sum of TP and false negative (FN) predictions.
F1 Score	It represents the harmonic mean of precision and recall, striking a balance between these metrics. Particularly useful for evaluating models with imbalanced datasets, the F1 Score accounts for both false positives and false negatives in its calculation: 2 × Precision × Recall/(Precision + Recall).[16]

RESULTS

1. Descriptive Analysis

After preprocessing the data, which involved cleaning and replacing columns such as "NObeyesdad" with calculated BMI values for each individual, a kernel density estimate (KDE) and boxplot for BMI are presented in this Figure. The dataset comprises 34,598 rows and 18 columns. Apart from the "NObeyesdad" column, there were no missing values present. Following the replacement of the "NObeyesdad" column with the "bmi" column, no additional missing data were identified in the dataset.



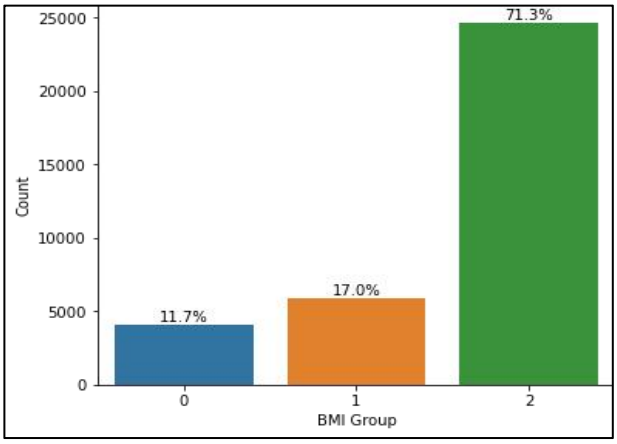


Based on the BMI values in the "bmi" column, individuals were classified into four categories: Underweight (4,061 individuals, 11.7%), Normal (5,769 individuals, 16.7%), Overweight (7,796 individuals, 22.5%), and Obese (16,972 individuals, 49.1%). As the Figure depicts, the synthetic data originated mostly from participants with obesity, with approximately 72% of the participants classified as overweight or obese, while only about 17% were categorized as normal.

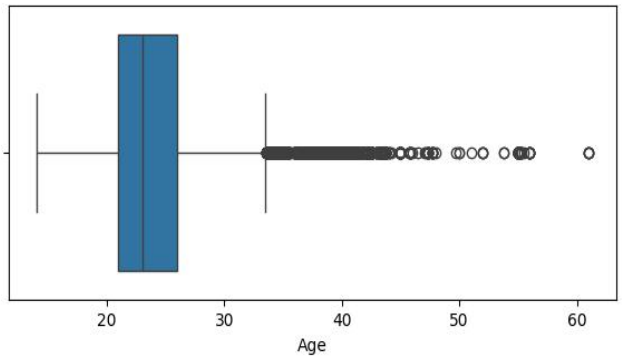
For further predictive analysis, individuals were categorized into three groups based on their BMI: underweight (BMI Group = 0),

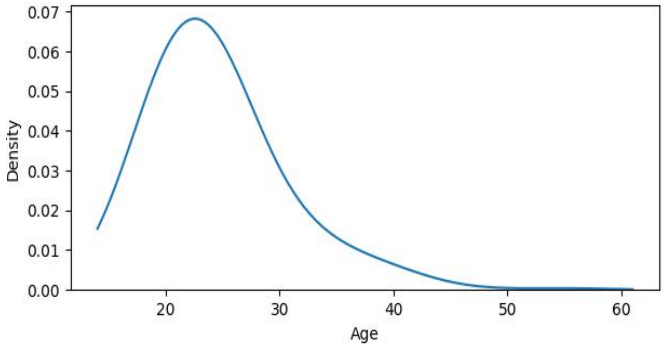
normal (BMI Group = 1), and overweight or obese (BMI Group = 2).

The BMI Group of 1 corresponds to the normal group and BMI Group of 2 represents the overweight or obese groups Then, the underweight group was excluded from the predictive analysis. Accordingly, approximately 81% of individuals were classified as overweight or obese, constituting four times the number in the normal group. After excluding underweight individuals from the dataset, the number of rows reduced to 30,537.



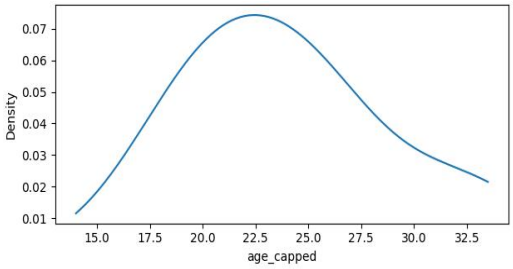
Outliers were identified in the Age feature through boxplots, z-scores or standard deviation, and the interquartile range (IQR), where data points outside the range of Q1 - 1.5 * IQR and Q3 + 1.5 * IQR were considered as outliers. According to the box plot, the data points which are outside 1.5 times the IQR below Q1 and above Q3 were considered as outliers.

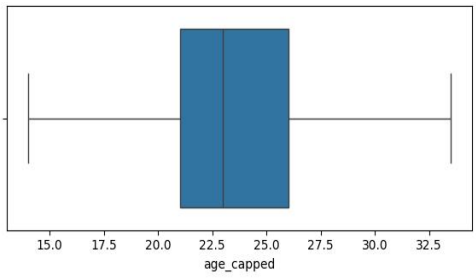




As the KDE plot of the age variable reveals outliers are located on the right-skewed side of the graph. Considering the importance of age as a critical risk factor for obesity, addressing outliers is crucial to mitigate potential biases in the results.

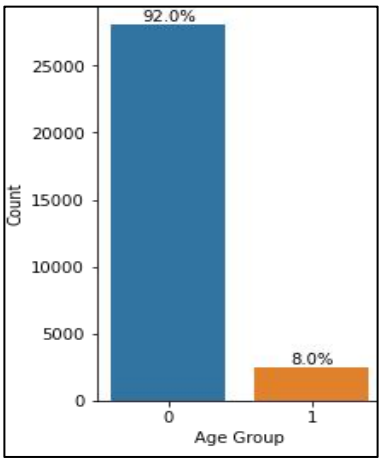
The **Trimming** and **Analysis** of outliers were conducted as a means of outlier treatment. Initially, outliers were identified as data points lying beyond 3 SDs from the mean, encompassing values both above and below the mean and then they were capped. The KDE plot of the age variable provides a visualization of the outcomes following the trimming or removal of outliers, demonstrating a normal distribution of ages in both the KDE and box plots.





The data points falling within 3 SDs or with an age less than or equal to 33.5 were categorized as non-outliers, constituting 28,679 rows and 14 columns. Those data fell outside this range was deemed outliers. In normal group, 5,746 (18.8%) had an age less than or equal to 33.5, whereas in the overweight/obese group, this figure was higher, standing at 22,339 individuals (73.2%). The outliers with an age exceeding 33.5 were relatively rare in the normal group, comprising only 122 individuals (0.4%), whereas in the overweight/obese group, they were more prevalent, accounting for 2,330 individuals (7.6%).

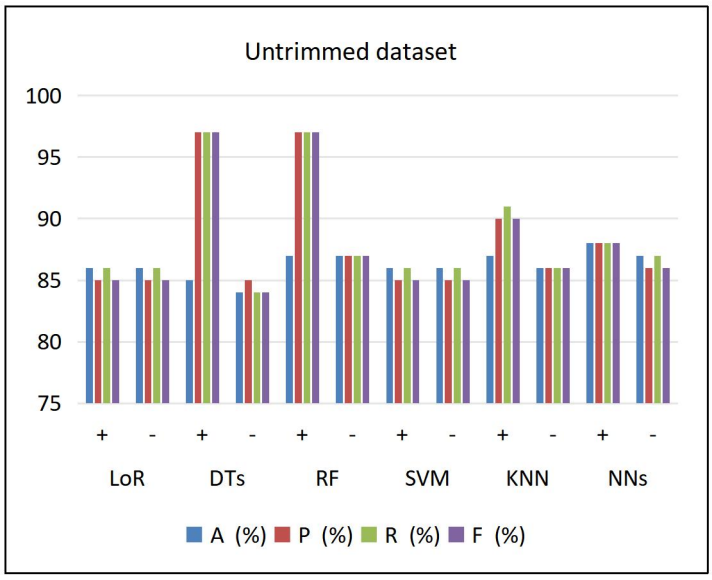
Next, following the identification of outliers exceeding 3 standard deviations from the dataset, the data underwent partitioning into two distinct groups: non-outliers (comprising 28,679 rows and 14 columns), consisting of individuals with an age below 33.5, and outliers (consisting of 1,858 rows and 14 columns), encompassing individuals aged 33.5 or older. This segregation facilitates further analysis aimed at assessing the influence of outliers on the outcomes. This is illustrated in the bar chart depicting the distribution of age categorized into non-outliers (Age Group = 0) and outliers (Age Group = 1). Individuals aged 33.5 or above are allocated to Age Group 1, while those below 33.5 are classified as Age Group 0. As depicted in this figure, 1,858 individuals (8% of the total) are identified as outliers due to their age exceeding 33.5. Given this notable disparity, coupled with the pivotal role of age as a risk factor for obesity, and recognizing the potential bias introduced by outliers, conducting further analysis becomes imperative to evaluate their impact on the overall findings.



Multicollinearity was assessed by examining the correlation between predictors. The analysis uncovered a strong correlation between height and weight (r = 0.5), as depicted in the provided Figure. The heatmap of the correlation matrix illustrates the presence of multicollinearity between height and weight. As height and weight were both used in calculating BMI, BMI was deemed a suitable replacement for height and weight in subsequent analyses, and height and weight were consequently excluded. Furthermore, given that the correlation coefficients for other predictors fell below the threshold, conducting further VIF analysis was deemed unnecessary.

2. Prediction

To examine the hypothesis, the performance of six supervised learning classification models—LoR, DTs, RF, SVM, KNN, and NNs—was assessed using accuracy, precision, recall, and F1 Score metrics. These metrics were compared across datasets with and without age outliers, employing both scikit-learn's train-test split (CV^-) and 10-fold cross-validation (CV^+) methods for each model as summarized in Table 4.



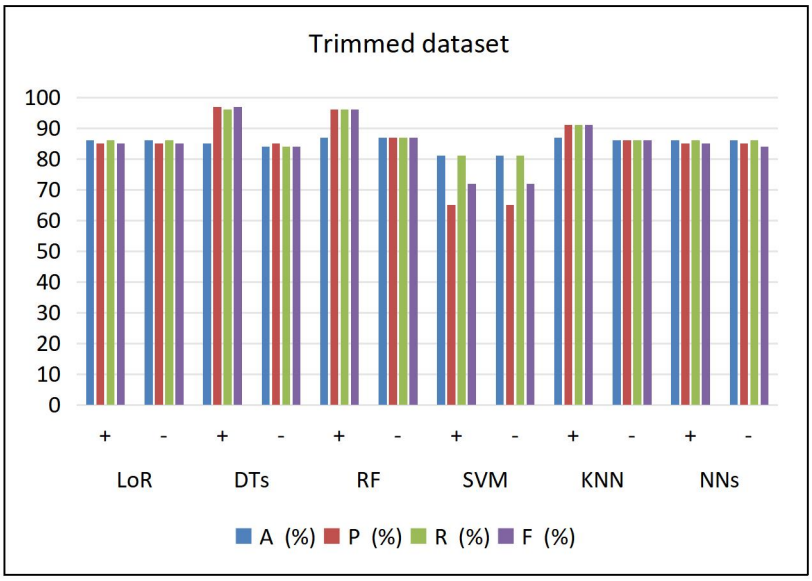
According to this table, the performance metrics for all models ranged from 72% to 97%, spanning a distance of 25%. This variability highlights the diverse performance levels exhibited by the classifiers. Examining the untrimmed dataset in this Figure showcases the variability of performance metrics, among the six classifiers, DTs and RF exhibited the most promising performance for CV^+, achieving 97% precision, recall, and F1-score for both models. This suggests that DTs and

RF were highly adept at accurately classifying individuals across various risk categories of obesity, demonstrating their robustness in capturing complex patterns within the data.

Conversely, DTs showed comparatively lower performance when 10-fold CV was not utilized (CV^-), with precision, recall, and F1-score metrics at 85%, 84%, and 84%, respectively. This disparity in performance underscores the importance of employing robust evaluation methodologies like CV^+, which can mitigate overfitting and provide more reliable estimates of model performance. Furthermore, the notable performance gap between models when employing different evaluation methods highlights the significance of selecting appropriate validation techniques tailored to the dataset characteristics and modeling objectives. It also underscores the need for thorough evaluation and comparison of classifiers to identify the most suitable approach for predicting obesity risk factors accurately.

Table 4. Predicting obesity risk involved employing various classifiers and assessing their performance based on accuracy, precision, recall, and F1 score metrics in the untrimmed and trimmed the age outliers with or without utilizing 10-fold cross-validation (CV) method.

Model	CV	Trimmed dataset				Untrimmed dataset			
		Aaccuracy (%)	Precision (%)	Recall (%)	F1 score (%)	Aaccuracy (%)	Precision (%)	Recall (%)	F1 score (%)
LoR	+	86	85	86	85	86	85	86	85
	-	86	85	86	85	86	85	86	85
DTs	+	85	97	96	97	85	97	97	97
	-	84	85	84	84	84	85	84	84
RF	+	87	96	96	96	87	97	97	97
	-	87	87	87	87	87	87	87	87
SVM	+	81	65	81	72	86	85	86	85
	-	81	65	81	72	86	85	86	85
KNN	+	87	91	91	91	87	90	91	90
	-	86	86	86	86	86	86	86	86
NNs	+	86	85	86	85	88	88	88	88
	-	86	85	86	84	87	86	87	86



In the trimmed dataset, the performance of all models showed improvement both CV^+ and CV^-, yet DTs and RF consistently emerged as the top-performing models. Even in this refined dataset, DTs and RF demonstrated superior predictive capabilities compared to other classifiers. Notably, SVM exhibited weaker performance than the other models, regardless of whether 10-fold CV was employed.

Furthermore, the performance of the remaining models, including LoR, KNN, and NNs, was relatively similar in both trimmed and untrimmed datasets. However, DTs and RF consistently outperformed them, underscoring their effectiveness in accurately predicting obesity risk factors.

Overall, DTs and RF models, particularly when using the CV+ method for random assignment of train and test sets, proved to be the most effective for both trimmed and untrimmed datasets. These models showcased robust performance, demonstrating their ability to generalize well to unseen data and effectively capture complex patterns within the dataset. In contrast, SVM exhibited the lowest performance specifically in the trimmed dataset, emphasizing the importance of selecting appropriate classifiers tailored to the dataset characteristics and modeling objectives.

DISCUSSION

The aim of this study was to evaluate the efficacy of six supervised learning classification models (LoR, DTs, RF, SVM, KNN, and NNs) in predicting the risk of obesity, while considering the impact of age outliers on model accuracy and CV by using accuracy, precision, recall, and F1 Score metrics across datasets with and without age outliers, employing both train-test split and 10-fold CV methods.

To achieve this objective, each classifier was assessed using different metrics. The evaluation was conducted under two scenarios: with and without age outliers, utilizing either a standard train-test split or a 10-fold CV method to partition the data into training and testing sets. This comprehensive approach facilitated a thorough analysis of how the presence of outliers and the utilization of CV techniques influenced the performance of the models.

The results, as presented in Table 4 and relevant figures, provide valuable insights into the behavior of each model under different conditions. The performance of all models varied from 72% to 97%, highlighting diverse performance levels across the classifiers. In both trimmed and untrimmed datasets, DTs and RF consistently outperformed other models, exhibiting 97% precision, recall, and F1-score in 10-fold CV. This indicates that their performance remains relatively stable regardless of the presence of age outliers, showcasing their robustness in capturing complex patterns.

Conversely, SVM demonstrated weaker performance compared to other models in the trimmed dataset, emphasizing the critical role of selecting appropriate classifiers tailored to dataset characteristics. Such variations in performance metrics when outliers are included in the dataset underscore the significance of their detection and removal in the preprocessing stage of ML tasks, particularly in scenarios where outliers can significantly influence model predictions.

Overall, this study underscores the significance of selecting suitable classifiers and robust evaluation methods tailored to dataset characteristics for optimal predictive modeling outcomes. The study's findings contribute to a better understanding of how different classifiers behave when tasked with predicting obesity risk, while also highlighting the impact of age outliers on model performance. These insights pave the way for future research aimed at improving predictive modeling in healthcare and other domains where outlier detection and mitigation are crucial.

REFERENCES

1. https://www.who.int/health-topics/obesity#tab=tab_1
2. https://www.worldobesity.org/resources/resource-library/world-obesity-atlas-2023
3. Jonathan Pearson-Stuttard et al. (2023): Real-world costs of obesity-related complications over eight years: a US retrospective cohort study in 28,500 individuals. International Journal of Obesity. 47:1239–1246.
4. https://data.worldbank.org/indicator/NY.GDP.MKTP.CD
5. World Health Organization. BMI. https://www.who.int/data/gho/data/themes/topics/topic-details/GHO/body-mass-index?introPage=intro_3.html. Accessed 12 Dec. 2023.
6. Zhou XB, Chen L, Liu HX (2022): Applications of Machine Learning Models to Predict and Prevent Obesity: A Mini-Review. Front Nutr 9:933130.
7. Reade W and Chow A (2024): Multi-Class Prediction of Obesity Risk. Kaggle. https://kaggle.com/competitions/playground-series-s4e2

8. Palechor M and Hoz Manotas A (2019): Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and MexicoFabio. Data Brief 2:25:104344.
9. https://www.cdc.gov/healthyweight/assessing/bmi/index.html
10. Chan JY, Leow SMH, BKT, Cheng WK, Phoong SW, Hong ZW, Chen YL (2022). Mitigating the Multicollinearity Problem and Its Machine Learning Approach: A Review. Mathematics: 10(8), 1283.
11. https://web.stanford.edu/~jurafsky/slp3/5.pdf
12. Bansal M, Goyal A, Choudhary A (2022). A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning. Decision Analytics Journal. 3:100071.
13. Rodriguez-Galiano V, Sanchez-Castillo M, Chica-Olmo M, Chica-Rivas M (2015). Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. Ore Geology Reviews: 71: 804-818.
14. Blanco-Justicia A & Domingo-Ferrer J (2019). Machine Learning Explainability Through Comprehensible Decision Trees. International Cross-Domain Conference for Machine Learning and Knowledge Extraction: 15–26.
15. Han SH, Kim KW, Kim SY, Yound YC (2018). Artificial Neural Network: Understanding the Basic Concepts without Mathematics. Dement Neurocogn Disord. 17(3): 83–89.
16. Hastie T, Tibshirani R, Friedman J (2008). The Elements of Statistical Learning. Springer Series in Statistics.

APPENDIX

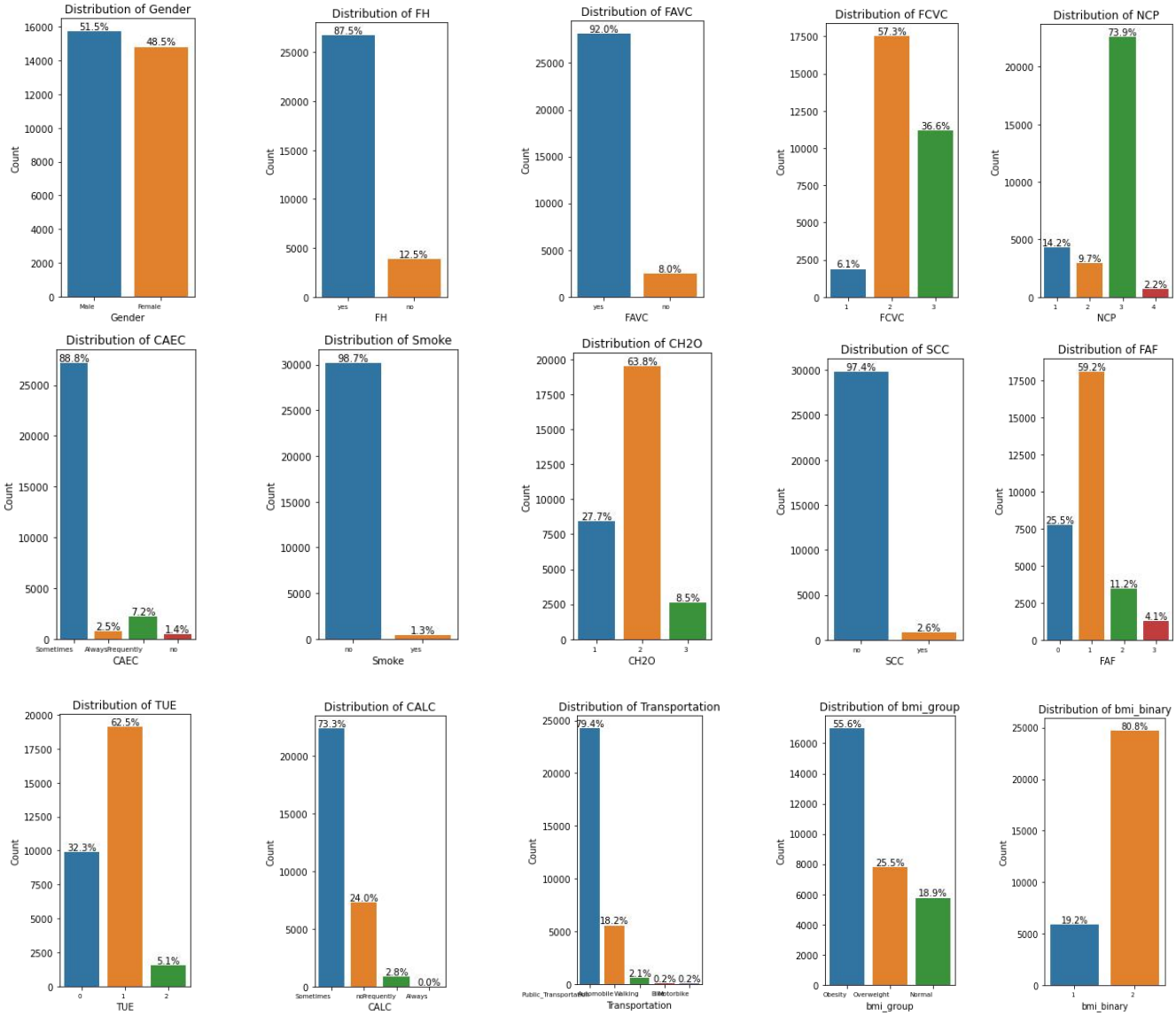


Figure 1. Distribution of different features.

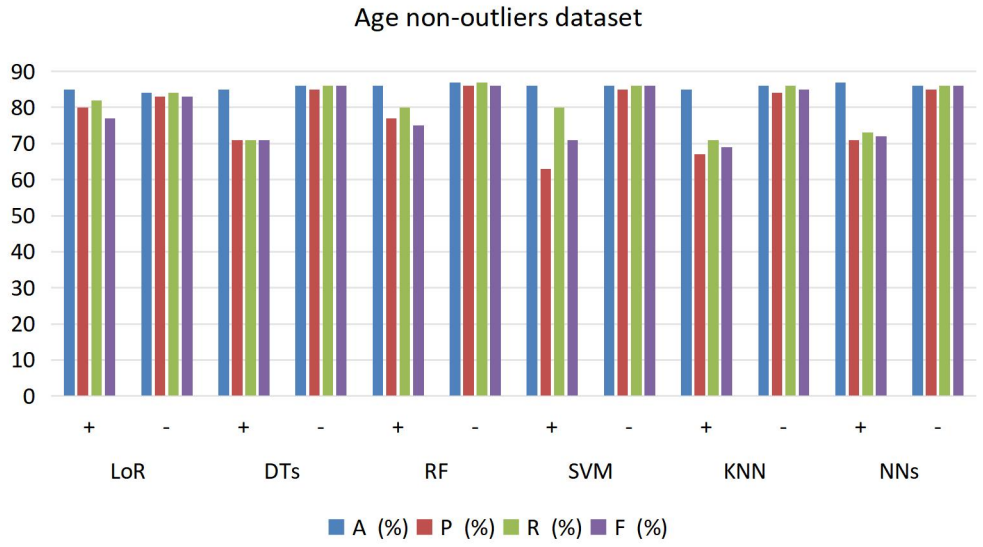


Figure 2. Demonestration of perofmance of classifiers by accuracy (A, precision (P), recall (R), and F1 Score (F) metrics across non-outliers of age data employing both train-test split and 10-fold CV methods.

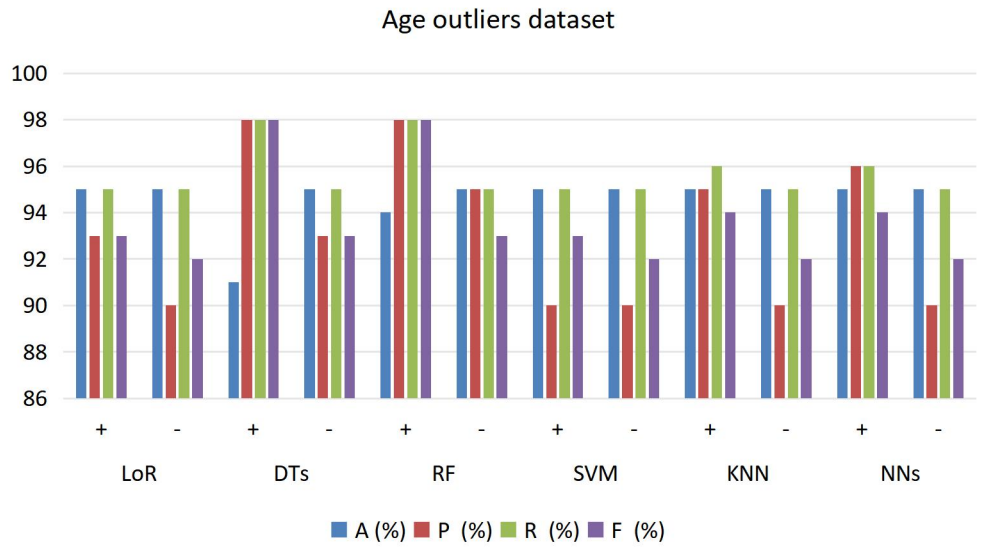


Figure 3. Demonestration of perofmance of classifiers by accuracy (A, precision (P), recall (R), and F1 Score (F) metrics across outliers of age data employing both train-test split and 10-fold CV methods.

Table 5. Confusion matrix of classifiers across untrimmed and trimmed data employing train-test split method.

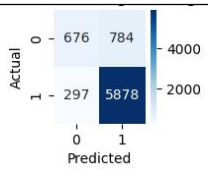
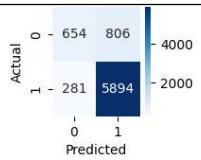
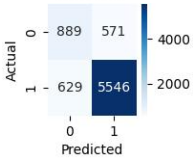
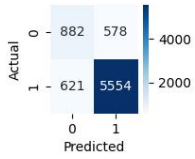
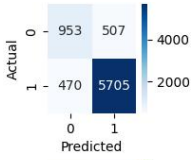
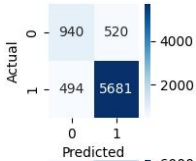
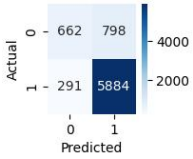
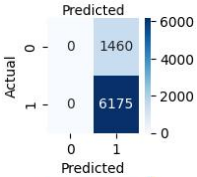
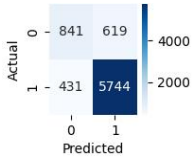
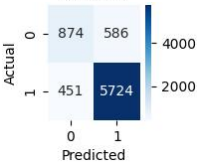
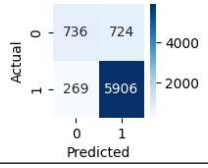
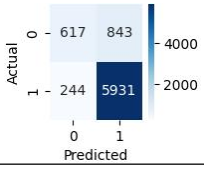
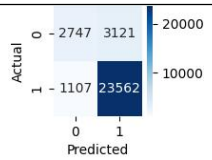
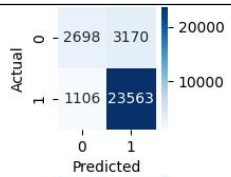
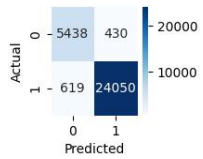
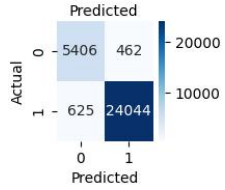
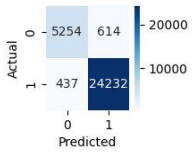
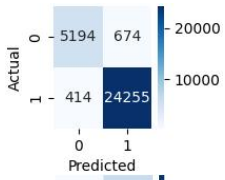
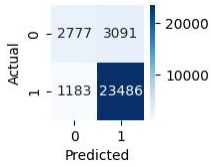
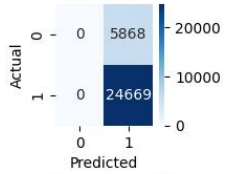
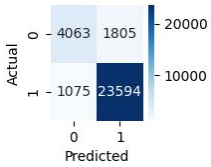
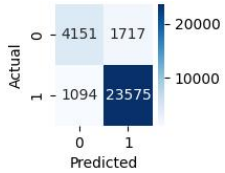
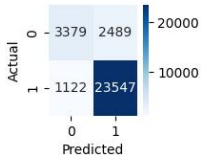
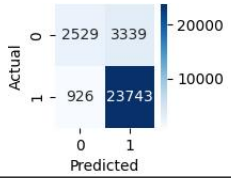
Model	Confusion Matrix	
	Untrimmed dataset	Trimmed dataset
LoR		
DTs		
RF		
SVM		
KNN		
NNs		

Table 6. Confusion matrix of classifiers across untrimmed and trimmed data employing 10-fold CV method.

Model	Confusion Matrix	
	Untrimmed dataset	Trimmed dataset
LoR		
DTs		
RF		
SVM		
KNN		
NNs		

CODE

```
# Library
import zipfile
from IPython.display import Image, display
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.exceptions import ConvergenceWarning
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, mean_squared_error, f1_score,
from sklearn.model_selection import cross_val_score
classification_report, recall_score
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings

"""
Data:
1. Attributes
* Gender, Age, Height, and Weight: physical attributes
* family_history_with_overweight: points out to familiarity with obesity
* FAVC: Frequent consumption of high caloric food
* FCVC: Frequency of consumption of vegetables
* NCP: Number of main meals
* CAEC: Consumption of food between meals
* SMOKE: tobacco usage
* CH20: Consumption of water daily
* SCC: Calories consumption monitoring
* FAF: Physical activity frequency
* TUE: Time using technology devices
* CALC: Consumption of alcohol
* MTRANS: Transportation used

2. Outcome
* NObeyesdad: Target
"""
# Read multiple CSV files from the ZIP archive: kaggle competitions download -c playground-series-s4e2
with zipfile.ZipFile("playground-series-s4e2.zip") as z:
    csv_files = [f for f in z.namelist() if f.endswith('.csv')][:3]  # Get the first 3 CSV files
    with z.open(csv_files[0]) as file1, z.open(csv_files[1]) as file2, z.open(csv_files[2]) as file3:
        x1 = pd.read_csv(file1)
        x2 = pd.read_csv(file2)
        x3 = pd.read_csv(file3)
```

```
df = pd.concat([x2, x3], ignore_index=True)

print(f'Shape of sample_submission.csv = {x1.shape}, train.csv = {x3.shape}, test.csv = {x2.shape}.')
print()
print(f'The type of df is {type(df)} and its shape = {df.shape}.')
print()
df.head().T
```

1. Data Preprocessing

Data Cleaning

```
# Rename column names in DataFrame df
df = df.rename(columns={'family_history_with_overweight': 'FH', 'MTRANS': 'Transportation','SMOKE':'Smoke'})
del df['NObeyesdad']
df.columns

# Add BMI_number column
df['bmi'] = df['Weight']/(df['Height']**2) # formula: weight/height^2
df['bmi'].head().T

# Add BMI category column
def bmi_grp(row):

    bmi = row['bmi']
    if bmi <= 18.5:
        return 'Underweight'
    elif 18.5 < bmi <= 24.9:
        return 'Normal'
    elif 25.0 <= bmi < 29.9:
        return 'Overweight'
    else:
        return 'Obesity'

df['bmi_group'] = df.apply(bmi_grp, axis=1)
df[['bmi','bmi_group']].head().T

plt.figure(figsize=(6, 5))
ax = sns.countplot(x='bmi_group', data=df)#, color='darkblue'
plt.xlabel('BMI Group')
plt.ylabel('Count')

# Add table of frequency (percentage) at the top of each column
total = len(df['bmi_group'])
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height() / total)
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    ax.annotate(percentage, (x, y), ha='center', va='bottom')

plt.show()

# Update Features
df1 = df.copy()
```

```
df1['bmi_binary'] = df1['bmi'].apply(lambda x: 0 if x <= 18.5 else (1 if x <= 25 else 2))
df1['FCVC'] = df1['FCVC'].apply(lambda x: 1 if 0 < x < 1 else (1 if 1 < x < 2 else (2 if 2 < x < 3 else (4 if 4 < x < 5 else (5 if 5 < x < 6 else (6 if 6 < x < 7 else x))))))).astype("int").astype("category")
df1['NCP'] = df1['NCP'].apply(lambda x: 1 if 0 < x < 1 else (1 if 1 < x < 2 else (2 if 2 < x < 3 else (4 if 4 < x < 5 else (5 if 5 < x < 6 else (6 if 6 < x < 7 else x))))))).astype("int").astype("category")
df1['FAF'] = df1['FAF'].apply(lambda x: 1 if 0 < x < 1 else (1 if 1 < x < 2 else (2 if 2 < x < 3 else (4 if 4 < x < 5 else (5 if 5 < x < 6 else (6 if 6 < x < 7 else x))))))).astype("int").astype("category")
df1['TUE'] = df1['TUE'].apply(lambda x: 1 if 0 < x < 1 else (1 if 1 < x < 2 else (2 if 2 < x < 3 else (4 if 4 < x < 5 else (5 if 5 < x < 6 else (6 if 6 < x < 7 else x))))))).astype("int").astype("category")
df1['CH2O'] = df1['CH2O'].apply(lambda x: 1 if 0 < x < 1 else (1 if 1 < x < 2 else (2 if 2 < x < 3 else (4 if 4 < x < 5 else (5 if 5 < x < 6 else (6 if 6 < x < 7 else x))))))).astype("int").astype("category")

categorical_features = df1.columns[df1.dtypes=="object"].tolist()
numeric_features = df1.columns[df1.dtypes!="object"].tolist()

print(categorical_features)
print(numeric_features)
print(df1[categorical_features].nunique())
print(df1[numeric_features].nunique())

df1[['bmi_binary','FCVC', 'NCP', 'FAF', 'TUE', 'CH2O']].info()
df1.shape

plt.figure(figsize=(6, 5))
ax = sns.countplot(x='bmi_binary', data=df1)#, color='darkblue'
plt.xlabel('BMI Group')
plt.ylabel('Count')

# Add table of frequency (percentage) at the top of each column
total = len(df1['bmi_binary'])
for p in ax.patches:
   percentage = '{:.1f}%'.format(100 * p.get_height() / total)
   x = p.get_x() + p.get_width() / 2
   y = p.get_height()
   ax.annotate(percentage, (x, y), ha='center', va='bottom')

plt.show()

# Target: bmi binary
df1 = df1[df1['bmi_binary'] != 0]
df1.shape

plt.figure(figsize=(6, 5))
ax = sns.countplot(x='bmi_binary', data=df1)#, color='darkblue'
#plt.title('Distribution of BMI Groups')
plt.xlabel('BMI Group')
plt.ylabel('Count')

# Add table of frequency (percentage) at the top of each column
total = len(df1['bmi_binary'])
for p in ax.patches:
   percentage = '{:.1f}%'.format(100 * p.get_height() / total)
   x = p.get_x() + p.get_width() / 2
```

```
    y = p.get_height()
    ax.annotate(percentage, (x, y), ha='center', va='bottom')

plt.show()

## Missing Data
# Check for missing values in DataFrame df
if df1.isnull().values.any():
    print("There are missing values in the DataFrame.")
else:
    print("No missing values found in the DataFrame.")

df1.isnull().sum()

## Outliers
# Trimming approach
# 1. Outlier Detection using Z-score
# Calculate Z-scores for the 'Age' column
z_scores = np.abs(stats.zscore(df1['Age']))

# Define a threshold for identifying outliers (e.g., Z-score > 3)
outlier_threshold = 3

# Identify outliers based on Z-scores
outliers = df1['Age'][z_scores > outlier_threshold]

# 2. and Treatment  Treat outliers by capping/extending the range
lower_bound = df1['Age'].quantile(0.25) - 1.5 * (df1['Age'].quantile(0.75) - df1['Age'].quantile(0.25))
upper_bound = df1['Age'].quantile(0.75) + 1.5 * (df1['Age'].quantile(0.75) - df1['Age'].quantile(0.25))

# Cap outliers beyond the lower and upper bounds
df1['age_capped'] = df1['Age'].clip(lower=lower_bound, upper=upper_bound)

# Display the updated DataFrame with capped outliers
print(df1[['Age', 'age_capped']].head().T)
print(np.max(df1['age_capped']))

# Create a figure with subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 6))

# Plot boxplot for 'Age' in the first row
sns.boxplot(x=df1["Age"], ax=axs[0, 0])
sns.kdeplot(data=df1, x="Age", bw_adjust=5, cut=0, ax=axs[0, 1])

# Plot boxplot for 'Age_capped' in the second row
sns.boxplot(x=df1["age_capped"], ax=axs[1, 0])
sns.kdeplot(data=df1, x="age_capped", bw_adjust=5, cut=0, ax=axs[1, 1])

plt.tight_layout()
plt.show()

# 2. Outlier analysis approach
df1['age_group'] = np.where(df1['Age'] <= 33.5, 0, 1)
```

```
# Count for each category
count_age0_normal = len(df1[(df1['age_group'] == 0) & (df1['bmi_binary'] == 1)])
count_age0_overweight = len(df1[(df1['age_group'] == 0) & (df1['bmi_binary'] == 2)])
count_age1_normal = len(df1[(df1['age_group'] == 1) & (df1['bmi_binary'] == 1)])
count_age1_overweight = len(df1[(df1['age_group'] == 1) & (df1['bmi_binary'] == 2)])

# Create a table to display the counts
table_data = {'Age < 33.5 & Normal': [count_age0_normal],
        'Age < 33.5 & Overweight/Obese': [count_age0_overweight],
        'Age >= 33.5 & Normal': [count_age1_normal],
        'Age >= 33.5 & Overweight/Obese': [count_age1_overweight]}

# Display the table
D = pd.DataFrame(table_data)
print(D.T)

plt.figure(figsize=(3, 5))
ax = sns.countplot(x='age_group', data=df1)#, color='darkblue'
plt.xlabel('Age Group')
plt.ylabel('Count')

# Add table of frequency (percentage) at the top of each column
total = len(df1['age_group'])
for p in ax.patches:
   percentage = '{:.1f}%'.format(100 * p.get_height() / total)
   x = p.get_x() + p.get_width() / 2
   y = p.get_height()
   ax.annotate(percentage, (x, y), ha='center', va='bottom')

plt.show()

## Multicollinearity
# Calculate the correlation matrix
corr_matrix = df1[['Age',
            #'age_capped',
            #'age_group',
            'Height',
            'Weight',
            'FCVC',
            'NCP',
            'CH2O',
            'TUE',
            ]].corr()

# Create a heatmap to visualize the correlation matrix
plt.figure(figsize=(12, 4))
ax = sns.heatmap(corr_matrix, annot=True, cmap="Blues", fmt=".1f")
ax.set(xlabel="", ylabel="")
ax.xaxis.tick_top()
plt.show()

### 2. Descriptive Analysis
```

```
def bar_charts(DF, target):

    for column in target:
        plt.figure(figsize=(3, 5))
        ax = sns.countplot(x=column, data=DF)

        plt.title(f'Distribution of {column}')
        plt.xlabel(column)
        plt.ylabel('Count')

        ax.set_xticklabels(ax.get_xticklabels(), rotation=0, ha='right', fontsize=7)
        plt.tight_layout()

        total = len(DF[column])
        for p in ax.patches:
          percentage = '{:.1f}%'.format(100 * p.get_height() / total)
          x = p.get_x() + p.get_width() / 2
          y = p.get_height()
          ax.annotate(percentage, (x, y), ha='center', va='bottom')

        plt.show()

# Call the function with the DataFrame df1 and the target column 'Obesity Risk Levels'
columns = ['age_group', 'Gender', 'FH', 'FAVC', 'FCVC', 'NCP', 'CAEC', 'Smoke', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC',
'Transportation', 'bmi_group', 'bmi_binary']
bar_charts(df1, columns)

### 3. Prediction & Visualization
# Encoding the categorical data
columns_to_encode = ['Gender', 'age_group', 'age_capped', 'FH', 'FAVC', 'CAEC', 'Smoke', 'SCC', 'CALC', 'Transportation',
'bmi_binary']

label_encoder = LabelEncoder()

# Apply label encoding using a for loop
for column in columns_to_encode:
    df1[column] = label_encoder.fit_transform(df1[column])

## Modeling
# 1. Classification without CV
def Classification(DF):

  warnings.filterwarnings("ignore", category=ConvergenceWarning)

  X = DF.drop(['bmi_binary'], axis=1)
  y = DF['bmi_binary']

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

  models = {'Logistic Regression': LogisticRegression(max_iter = 1000),
          'Decision Trees': DecisionTreeClassifier(),
          'Random Forest': RandomForestClassifier(),
          'Support Vector Machine': SVC(),
```

```
        #'Kernel SVM model': SVC(kernel='rbf'),
        'K-Nearest Neighbors': KNeighborsClassifier(),
        'Neural Networks': MLPClassifier(alpha=1, hidden_layer_sizes=(20, 10))}

 # Train and evaluate each model
 model_names, accuracies, precisions, recalls, f_scores = [], [], [], [], []

 for name, model in models.items():

   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)

   accuracy = accuracy_score(y_test, y_pred)
   confusion_mat = confusion_matrix(y_test, y_pred)
   precision = precision_score(y_test, y_pred, average='weighted')
   recall = recall_score(y_test, y_pred, average='weighted')
   f_score = f1_score(y_test, y_pred, average='weighted')

   model_names.append(name)
   accuracies.append(accuracy)
   precisions.append(precision)
   recalls.append(recall)
   f_scores.append(f_score)

   # Plot confusion matrix
   plt.figure(figsize=(1.5, 1.5))
   sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
   plt.xlabel('Predicted')
   plt.ylabel('Actual')
   plt.title(f'Confusion Matrix - {name}')
   plt.show()

 # Create a DataFrame to store the evaluation metrics
 metrics_df = pd.DataFrame({'Model': model_names,'Accuracy': accuracies,'Precision': precisions,'Recall': recalls,'F1 Score':f_scores})

 # Display the metrics table
 print(metrics_df)

# 2. Classification with CV
def Classification_CV(DF):

   warnings.filterwarnings("ignore", category=ConvergenceWarning)

   X = DF.drop(['bmi_binary'], axis=1)
   y = DF['bmi_binary']

   models = {'Logistic Regression': LogisticRegression(max_iter=1000),
         'Decision Trees': DecisionTreeClassifier(),
         'Random Forest': RandomForestClassifier(),
         'SVM': SVC(),
         'K-Nearest Neighbors': KNeighborsClassifier(),
         'Neural Networks': MLPClassifier()}
```

```
    # Initialize lists to store evaluation metrics
    model_names, accuracies, precisions, recalls, f_scores = [], [], [], [], []

    for name, model in models.items():
        # Perform 10-fold cross-validation
        cv_scores = cross_val_score(model, X, y, cv=10, scoring='accuracy')

        # Calculate mean accuracy across folds
        accuracy = np.mean(cv_scores)

        # Fit the model on the entire dataset
        model.fit(X, y)
        y_pred = model.predict(X)

        # Calculate other evaluation metrics
        confusion_mat = confusion_matrix(y, y_pred)
        precision = precision_score(y, y_pred, average='weighted')
        recall = recall_score(y, y_pred, average='weighted')
        f_score = f1_score(y, y_pred, average='weighted')

        model_names.append(name)
        accuracies.append(accuracy)
        precisions.append(precision)
        recalls.append(recall)
        f_scores.append(f_score)

        # Plot confusion matrix
        plt.figure(figsize=(1.5, 1.5))
        sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title(f'Confusion Matrix - {name}')
        plt.show()

    # Create a DataFrame to store the evaluation metrics
    metrics_df = pd.DataFrame({'Model': model_names, 'Accuracy': accuracies, 'Precision': precisions, 'Recall': recalls, 'F1 Score': f_scores})

    # Display the metrics table
    print(metrics_df)

# Call the functions: data
df_Age = df1[['Gender','Age','FH','FAVC','FCVC','NCP','CAEC','Smoke','CH2O','SCC','FAF','TUE','CALC','Transportation','bmi_binary']]
df_age_capped = df1[['Gender','age_capped','FH','FAVC','FCVC','NCP','CAEC','Smoke','CH2O','SCC','FAF','TUE','CALC','Transportation','bmi_binary']]

# Call the functions: Untrimmed/Trimmed data without CV
Classification(df_Age)
Classification(df_age_capped)

# Call the functions: Untrimmed/Trimmed data with CV
Classification_CV(df_Age)
Classification_CV(df_age_capped)
```