TECNOLÓGICO DE MONTERREY

Department of Engineering and Science

PhD in Computational Science

Mathematics for Computer Science

Group: 550

Statistics

Estimation of Obesity Levels Based on Eating Habits and Physical Condition

Submission date: November 15th, 2025

Professors:

Dr. Jorge Isaac Chairez Oria
Dr. Raúl Monroy Borja
Dr. José Gerardo Tamez Pena

Student:

Maria Camila Castaño Martínez

Contents

Mathematics for Computer Science
Statistics

María Camila Castaño Martínez

November 15th, 2025

1 Introduction

Every table, algorithm and figure can be found in the Annexes section of the report.

This report analyzes a donated dataset containing information on obesity levels among participants from Mexico, Peru and Colombia, based on their eating habits and physical activities (condition too). The dataset includes 2111 participants and 16 variables that cover factors related to their lifestyle. Each variable was classified according to measurement level. This data was extracted from the UCI Machine Learning Repository [2]. At first sight, the dataset contains a mix of **categorical, integer, binary and continuous (float)** variables. All variable names were identified with "*estimation_of_obesity_levels _based_on_eating_habits_and_physical_condition*" variable section, and the data were found to be complete, with no missing values. For continuous variables, it was used distributional levels; ordinal variables were primarily treated as discrete variables for ordinal-aware analyses, and binary variables were encoded as 0/1 when needed. These choices guided the choice of statistical tests: ANOVA and KDE for continuous variables, chi-square / Fisher exact tests for categorical variables, and ordinal-aware methods (or rank-based methods) where appropriate. The only modification made was the simplification of the categorical variable *NObeyes*.

Based on the information from the dataset, we can agree for the research question to be: **'How do eating habits and physical activity patterns influence obesity levels among individuals?'**

From this general research question and based on the main 6 tasks related to statistical analysis, there are certain general objectives to complete:

- Describe the distributional behavior of all features, conditioned on obesity levels (target).
- Statistically assess how eating habits and physical activity differ across obesity levels using hypothesis testing for continuous, binary and categorical variables.

- Evaluate the discriminative ability of each feature by identifying which behaviors and physical activity patterns best predict obesity levels.

2 Materials and Methods (Theoretical Background)

This analysis was developed on Google Colab as a notebook in the Python language, fairly well known in the data analysis field. Luckily, the dataset used from the repository was complete and clean. The packages used for the analysis development can be found on Annexes (Algorithm 3).

Now, for the statistical development during the analysis, the following methods were used (depending on the task):

2.1 Probability Mass Function (PMF)

A PMF is a known mathematical function that can calculate the probability that a discrete (in this case, integer) random variable will have a specific value. It is used to describe the possibility of each outcome in a discrete probability distribution [5]. PMFs can be represented by a table or a graph and are fundamental for calculating measures like expected value and variance. For a discrete random variable, each possible value must have a non-zero likelihood.

The standard notation for a probability mass function is:

$$P(X = x) = f(X)$$

Where:

- X is the discrete random variable
- x is one of the possible discrete values
- $f(x)$ is a mathematical function that calculates the likelihood for the value of x

  Meaning, the chance of variable x assuming the specific value of x equals $f(x)$.

2.2 Probability Density Function (PDF)

This function is a statistical tool that describes the likelihood of a continuous random variable falling within a specific range. Does not give the probability of a single value (which is zero for a continuous variable), but the probability of the variable falling within the interval. It is needed to integrate the function over the interval to find this probability [6]. The area under its curve between 2 points represents the probability that the variable will take a value in that interval, its value is always

non-negative. Also, the total area under the curve over its entire range is equal to 1. Here's the known equation.

$$f(x) = \frac{d}{dx}F(x),$$

$F(x)$ is the cumulative distribution function. For Kernel Density Estimation from empirical PDF:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

Actually, the CDF is closely related to the PDF. This last one give the probability density and the CDF gives the probability that the random variable is less than or equal to a specific value. More information of this function on the next subsection.

2.3 Cumulative Distribution Function (CDF)

The CDF is a fundamental concept in probability theory that provides a way to describe the distribution of the random variable. Its a non-decreasing function that ranges from 0 to 1 capturing the entire probability distribution of the random variable. Represents the probability that a random variable has a value less than or equal to a certain value [3].

The Cumulative Distribution Function $F(x)$ of the random variable X is defined as:

$$F(x) = P(X \leq x)$$

2.4 Analysis of Variance (ANOVA)

ANOVA is a statistical method used to compare the means of 3 or more independent groups to determine if there are statistically significant differences among them [1]. It works by comparing the variance between group means with the variance within each group, using an F-statistic to determine if the variation between groups is larger than what is expected by chance.

The test statistic for ANOVA, which is the ratio of the variance between groups to the variance within groups:

$$F = \frac{variance\ between\ groups}{variance\ within\ groups}$$

2.5 False Discovery Rate (FDR)

It is a statistical method used in multiple hypothesis testing to control the rate of false positives. Defined as the expected proportion of incorrectly rejected null hypothesis (false discoveries) among all hypotheses rejected. It is useful in fields such as genomics where researchers need to identify a large number of potential discoveries while minimizing false positives (Type I errors) [7].

The FDR is defined:

$$FDR = E\left[\frac{V}{R}\right]$$

2.6 Receiver Operating Characteristic (ROC)

ROC (Receiver Operating Characteristic) analysis is a statistical tool that evaluates the diagnostic accuracy of a test by visualizing the performance at all possible thresholds. It does this by plotting the true positive rate (sensitivity) against the false positive rate (1 - specificity) to create an ROC curve. Some of the key concepts involving this analysis are:

- True Positive Rate (Sensitivity): The proportion of actual positive cases that are correctly identified as positive.
- False Positive Rate (1 - Specificity): The proportion of actual negative cases that are incorrectly identified as positive.
- ROC Curve: A graph plotting the true positive rate against the false positive rate for different threshold values.

2.7 Bootstrap

Used as a resampling method that estimates statistics by repeatedly sampling a dataset with replacement to create multiple simulated samples [4]. Generates multiple simulated samples, facilitating the estimation of summary statistics, calculation of standard error, and execution of hypothesis tests.

As a final step, the independent variables were classified into 3 main types: Continuous, binary and categorical. At the beginning, there were 4 categories with 'Integer', but it was changed as some of the answers contain '.0' counting them as continuous values. The full organization can be seen on Table 1.2.

3 Results

The results and statistical analysis were organized in subsections. Each one of them will contain a sample of the tables and graphs produced on Google Colab, the complete tables, algorithms and graphs will be provided in the annexes section.

3.1 Marginal Distributions of Features

It was needed to estimate the Probability Mass Function (PMF) for all binary and categorical variables and the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) for continuous variable using bootstrap resampling for statistical accuracy.

In simple terms, the methods used were (algorithms can be found on Annexes Task 1):

- Bootstrap resampling: Appropriate method for proportion estimation and densities in real-world datasets (obesity levels on 3 countries).
- Empirical PMF for Discrete Variables: Used for discrete values representing non-numerical categories, and can also completely describe the distribution of these variables. Applied to binary variables (family history, FAVC, SMOKE, SCC) and categorical variables (Gender, CAEC, CALC, MTRANS, NObeyesdad).
- Kernel Density Estimation (KDE) for Continuous Variables.

For binary variables, it was possible to conclude based on its results strong irregularities in health and lifestyle behaviors. For example:

- Family history of overweight (Yes: **81.76%**) indicates strong familiar clustering of obesity risk.
- Smoking (SMOKE) (No **97.92%**), begin this feature nearly absent in the group.

Categorical variables were divided by Gender, Eating and Drinking Habits, and Transportation. These results present low alcohol and moderate snack consumption, and limited daily physical activity. Some examples of the results:

- Gender $\approx$ 50% **male,** 50% **female**
- CAEC (Eating between meals): **"Sometimes" - 83.61%**
- CALC (Alcohol consumption): **"Sometimes" - 66.37%**
- MTRANS (Transportation): **"Public Transportation" - 74.85%**

The complete results can be found on Annexes (Task 1).

3.2 Analysis of Continuous Features

In this task, the main focus was to understand how each continuous variable behaves across the four obesity categories (*Insufficient weight, Normal weight, Overweight, Obesity*). Divided by 3 steps, here are the results:

a) Conditional Means and Variances by Obesity Level

For this step, bootstrap method was used to obtain a reliable estimations.

With some of the results obtained during the development, it is possible to conduct an analysis based on each feature:

- Age: Variance is largest in **Overweight (54)** and **Normal Weight (25)**, indicating broader age distributions.
- Weight: Variances also grow with the obesity level, reflection broader physical condition in obese participants.
- Physical activity frequency (FAF): Presents a strong decreasing trend (*Insufficient weight: 1.25, Normal weight: 1.25, Overweight: 1.00, Obesity: 0.87*), indicating lower physical activity, causing an increment associated with obesity.

b) One-way ANOVA (test across obesity groups)

Use of OLS model.

To evaluate whether each continuous feature shows significant differences across the four obesity levels, its was conducted a one-way ANOVA using 2 equivalent paths:

- Regression-based approach (lm())
- Classical ANOVA formulation (aov())

This implementation ensures consistency across statistical frameworks. This division is shown:

1. ANOVA via lm():

$$Feature_i = \beta_0 + \beta_1 ObesityLevel_i + \varepsilon_i$$

2. ANOVA via aov(): In paralell with the same analysis, the classical ANOVA was computed so that there's a possible comparison between mean with the four obesity levels.

It's possible to see the complete results in Annexes Task 2. It shows the One-way ANOVA summary for continuous features with the respective assumptions.

c) Post-hoc (Welch's t-test + FDR correction)

The interpretation varies depending on the feature we focus, it is also known that these analyses depend in more than one feature. Here are some examples of these interpretations.

- FAF: Physical activity progressively decreases as obesity increases.
- Age: Only *Obesity vs Overweight* is not significant ($p = 0.22$), supporting *Insufficient Weight* $<$ *Normal* $<$ *Overweight* $\approx$ *Obesity*

3.3 Analysis of Binary and Categorical Features

First, it was estimated a conditional prevalence:

$$P(X = category \mid Obesity\ level)$$

To produce a set of proportions for every category within each obesity group. Some of these differ between the known levels. For example, **Obesity** has $N \approx 972$ while **Insufficient Weight** has $N \approx 272$. The resultant tables (Annex Task 3) serve as a description of the evidence before a formal hypothesis testing.

On the other hand, for the assessment of the prevalence of each feature across obesity levels it was used **Pearson's Chi-Square test of independence**. Appropriate for all variables (binary/categorical) and obesity levels that are nominal in this case.

By reading the binary and categorical values, it is possible to conclude a robust analysis of obesity levels related to features like *Family history of overweight, FAVC, SMOKE and SCC*. From this variables, it is possible assert some individuals with obesity overwhelmingly report a positive family history and that smoking is less common in the obesity category, but more common in lower-weight groups.

3.4 Conditional Distributions

It was needed to characterize how the distribution of each binary and categorical feature changes across the 4 obesity levels. Done by estimating the PMF and CDF for each level.

The conditional PMF was computed as:

$$\hat{P}(X = x \mid Obesity\ Level = g) = \frac{Count(X = x, g)}{Total\ observation(g)}$$

Corresponding exactly to **Algorithm 6** in the Annexes Task 1, applied without bootstrap resampling.

On the other hand, for each feature and obesity level, the CDF was obtained as the cumulative sum of the PMF values. Given the category order consistency across all groups.

Based on this development, it is possible to visualize in the graphs (Annex Task 4) PMF patterns related to eating habits. For example, **"Sometimes"** is the dominant category across all groups, especially in the **Obesity group** (98.15%), showing that people with obesity report mid-meal consumption at this frequency.

3.5 Model-Based Expectations for Continuous Features

The goal of this tas is to assess whether the conditional distributions obtained from *Kernel Density Estimation (KDE)* from Task 4. The table summarizes the comparison between:

- Mean Model: Integration of KDE-estimated conditional PDF.
- Mean Sample: Computed directly from observed data. (Task 2)
- Equivalent: Boolean indicator shows whether the model and sample based means differ, helping the determination of the KDE distribution as sufficiently close to empirical distribution.

Across the known features, most comparisons show small numerical differences between model-based and sample means. Only **Height (Obesity and Overweight)** fall within the tolerance threshold.

3.6 Discriminative Intervals and Bootstrap Evaluation

This final task was divided in 4 parts:

Identification of discriminative intervals $(a, b]$: Location of the intervals in each feature where a specific class is most over or under-represented.

Use of bootstrap sampling (n=1000 iterations): The use of 1000 iterations allowed the application of a test for a recorded decision, including metrics like False Positive Rate and CI bounds that were computed.

Type I error: Measures the probability that the interval incorrectly classifies individuals as belonging to a specific obesity level when they do not. For example:

- Extremely low: Age - Normal Weight, Weight - Insufficient Weight, FCVC - Normal Weight
- Moderately low: Height - Normal Weight (0.027), Weight - Obesity (0.045), NCP - Insufficient Weight (0.063)
- High and Very High type (>0.30): CH2O - Obesity (0.311), Age - Obesity (0.368), FCVC - Obesity (0.512)

Type II error

The power column measures the ability of the interval to correctly identify participants belonging to a specific class. The higher values (> 0.80) can indicate high sensitivity. For example:

- Excellent power (>0.90): Age - Insufficient Weight (0.993)
- Good Power (0.80-0.95): Weight - Obesity (0.844)

Finally, to evaluate how well each categorical feature can discriminate obesity levels it was needed to compute diagnostic metrics under a **one-vs-rest framework**. For each obesity category, the target was:

$$Y_{(lvl)} = \begin{cases} 1 & \textit{if instance belongs to level (lvl)} \\ 0 & \textit{otherwise} \end{cases}$$

Each feature was treated as a predictor, with the idea to compute results of: *Accuracy, Sensitivity, Specificity and Cohen's Kappa.*

4 Discussion

For this last section related to the analysis of the dataset provided, there is an explanation for the evaluation for the importance of having a diverse features and patients to have a full view of their lifestyle with the goal to predict different obesity levels, using power values and interval to qualify feature influence. Several features consistently showed high predictive power for obesity levels, only suggesting a strong potential of clinical or behavioral indicators. Most important of these are Age and Weight, extremely useful for several categories.

With FAVC and FAF features was possible to obtain evident patterns aligned with known obesity risk factors, aligned with clinical findings that explain that poor dietary choices and sedentary lifestyle

are major contributors to obesity.

However, there are some limitations with the dataset and methods reflected on the main analysis. The dataset includes synthetic augmentations that can alter variance and distributional shape shown in the results (Annexes); these kind of cases can under represent certain patterns that are complexe to observe and therefore, affect inferential analysis and statistics. On the other hand, by only using the variables Weight/Height to predict obesity introduces uncertainty and limitations to produce a correct diagnostic for the participants. Also, the class imbalance in some features like obesity categories, that contain fewer samples that can produce unstable estimations.

5 Conclusion

The complete analysis shows that obesity is influenced by a complex relationship between demographic and lifestyle factors, with weight, age, and meal frequency (NCP) being the strongest predictors, followed by diet-related behaviors (CH2O, FCVC) and physical activity measures (FAF, TUE).

All features were statistically significant, showing their collective contribution to understanding obesity risk. The methods used include F-statistics for effect size evaluation and statistical testing, with recommendations to extend the analysis using multivariate predictive models, and causal inference techniques to validate insights and explore interactions beyond associations. Some limitations include reliance on synthetic data, assumptions of normality, and a cross-sectional design, warranting cautious interpretation and future validation with longitudinal real-world data. This was said on the Discussion section.

6 References

References

[1] Cloud, S. G. (2019). Spotfire.

[2] Fabio Mendoza Palechor, A. D. l. H. M. (2019). Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Technical report, UC Irvine Machine Learning Repository.

[3] for Geeks, G. (2018). Cumulative distribution function.

[4] Frost, J. (2020a). Introduction to bootstrapping in statistics with an example.

[5] Frost, J. (2020b). Statistics by jim.

[6] Pishro-Nik, H. (2016). Introduction to probability, statistics and random processes.

[7] Rouam, S. (2013). *Encyclopedia of Systems Biology*.

7 Statement on the Ethical Use of AI

I used AI (Superblocks, ChatGPT), for code debugging, explanation of statistical terms and generation of LaTeX structure. The analysis of each task was produced by the author using Google Colab as the IDE.

8 Autoevaluation of Initial Submission

⚠ Acknowledged dataset limitations (sampling bias, multi-country origin, synthetic augmentation, uniform class distribution) and explicitly stated that conclusions apply only to a hypothetical population with the observed proportions. → *Partially completed with the intention to show limitations of the data without a profound explanation.*

⚠ Correctly classified every variable (nominal, binary, ordinal, continuous) with justification and discussed appropriate statistical treatment/encoding. → *Partially completed, only showed the variables, their types and ranges expected from the participants, the recommended methods for their analysis were added.*

✓ Presented descriptive statistics (means, variances/SD, empirical bounds) for all numeric variables stratified by obesity level, including at least one comparative table. → *It was redacted within the instructions, with the intention to be precise. The comparison was needed to analyze results and obtain insights.*

✓ Analyzed categorical predictors using frequency methods (ANOFA, chi-square, or contingency tables) with test statistics, df, and p-values. → *Completed within the instructions and needed after the implementation on Python.*

× Computed diagnostic metrics (accuracy, sensitivity, specificity, Cohen's kappa or weighted kappa) for key categorical/ordinal variables in a one-vs-rest framework. → *Wasn't aware of the needed comparison between levels, addition of new model to obtain results of ANOVA method. The task 2 was incomplete without this.*

⚠ Performed one-way ANOVA on numeric variables across obesity levels (via both lm() and aov()), checked assumptions, and reported results with post-hoc tests where applicable. → *It was needed to have the two-path implementation of ANOVA.*

✓ Estimated conditional PDFs/support intervals for numeric variables by obesity level (e.g., kernel density or quantiles). → *Completed and needed for the final analysis of the dataset.*

✓ Conducted ROC analysis with ROC curves, AUC values, significance testing (KS/DeLong), Youden's index thresholds, and FDR-adjusted p-values. → *Done and analyzed with the graphs on each feature and obesity level.*

✓ Maintained formal academic writing, proper LaTeX notation, clear tables/figures, logical structure, and methodological citations throughout. → *Done throughout all the document, it was a template from other institution modified.*

✓ Critically discussed limitations (e.g., circularity with Weight/Height → BMI, synthetic data effects, multiplicity, assumption violations) and suggested improvements. → *Final discussion and conclusion contains these limitations before and after the data treatment.*

9 Autoevaluation of Final Submission

✓ Acknowledged dataset limitations (sampling bias, multi-country origin, synthetic augmentation, uniform class distribution) and explicitly stated that conclusions apply only to a hypothetical population with the observed proportions. → *Changed.*

✓ Correctly classified every variable (nominal, binary, ordinal, continuous) with justification and discussed appropriate statistical treatment/encoding. → *Changed.*

✓ Presented descriptive statistics (means, variances/SD, empirical bounds) for all numeric variables stratified by obesity level, including at least one comparative table.

✓ Analyzed categorical predictors using frequency methods (ANOFA, chi-square, or contingency tables) with test statistics, df, and p-values.

✓ Computed diagnostic metrics (accuracy, sensitivity, specificity, Cohen's kappa or weighted kappa) for key categorical/ordinal variables in a one-vs-rest framework. → *Added. It wasn't present on the report (BEFORE).*

✓ Performed one-way ANOVA on numeric variables across obesity levels (via both lm() and aov()), checked assumptions, and reported results with post-hoc tests where applicable. → *Changed. Addition of two-path ANOVA*

✓ Estimated conditional PDFs/support intervals for numeric variables by obesity level (e.g., kernel density or quantiles).

✓ Conducted ROC analysis with ROC curves, AUC values, significance testing (KS/DeLong), Youden's index thresholds, and FDR-adjusted p-values.

✓ Maintained formal academic writing, proper LaTeX notation, clear tables/figures, logical structure, and methodological citations throughout.

✓ Critically discussed limitations (e.g., circularity with Weight/Height → BMI, synthetic data effects, multiplicity, assumption violations) and suggested improvements.

10 Annexes

The annexes are divided by the 6 proposed tasks in the following order:

- Task 1
- Task 2
- Task 3
- Task 4
- Task 5
- Task 6

Each of them contain, in order:

1. Algorithm (code snippet)
2. Table (results generated)
3. Graphs (if those were needed)

This with the intention of showing organized methods, implementation and results of the tasks.

In this next page, will be possible to consult the organization of the data, their categories, methods for each type and ranges.

Feature	**Type**	**Justification**	**Suggested Encoding / Treatment**
Age	Continuous	Numeric, measured on ratio scale	Keep numeric; KDE, ANOVA, standardize as needed
Height	Continuous	Numeric, measured on ratio scale	Keep numeric
Weight	Continuous	Numeric, measured on ratio scale; used to compute BMI (circularity)	Keep numeric; be explicit about BMI circularity
NCP (Number of main meals)	Ordinal	Discrete ordered categories (e.g., 1,2,3,4)	Treat as ordinal; consider integer encoding; KDE acceptable if many levels
FCVC (Veg consumption freq.)	Ordinal	Ordered frequency categories	Ordinal encoding; possibly treat as continuous only with caution
CH2O (Water intake)	Ordinal	Ordered frequency	Ordinal encoding
FAF (Physical activity freq.)	Ordinal	Ordered categories	Ordinal or integer encoding
TUE (Time using electronics)	Ordinal	Ordered or counts	Ordinal encoding; if many unique values, treat as continuous
fam_history_with_ow	Binary	yes/no	Map yes→1, no→0; use proportions, chi-square tests
FAVC (high-calorie food)	Binary	yes/no	Map yes→1, no→0
SMOKE	Binary	yes/no	Map yes→1, no→0
SCC (calorie counting)	Binary	yes/no	Map yes→1, no→0
Gender	Nominal	Male / Female; unordered	Dummy (one-hot) for models; report proportions
CAEC (consumption between meals)	Ordinal	categories like no / sometimes / frequently / always	Treat as ordinal; encode numeric increasing with frequency
CALC (alcohol consumption)	Ordinal	categories	Ordinal encoding
MTRANS (transportation)	Nominal	categories with no intrinsic order	One-hot encoding
NObeyesdad_new (target 4-class)	Ordinal*	Ordered levels from Insufficient → Normal → Overweight → Obesity	Use as categorical for tests; for one-vs-rest treat as binary; careful with ordinal tests

Table 1: Variable classification, justification, and recommended encoding strategies.

Feature	Possible values/Range
Gender	Female, Male
Age	14-61
Height (m)	1.45-1.98
Weight (kg)	39-173 aprox.
Family history with overweight	Yes/No
FAVC	Yes/No
FCVC	1-3
NCP	1-4
CAEC	'Sometimes', 'Frequently', 'Always', 'no'
SMOKE	Yes/No
CH2O	1-3
SCC	Yes/No
FAF	0-3
TUE	0-2
CALC	'no', 'Sometimes', 'Frequently', 'Always'
MTRANS	'Public Transportation', 'Walking', 'Automobile', 'Motorbike', 'Bike
NObeyes	'Insufficient Weight', 'Normal Weight', 'Obesity', 'Overweight'

Table 2: Variables and expected answers from patients.

```
# recode obesity:
obe = {'Insufficient_Weight': 'Insufficient Weight',
       'Normal_Weight': 'Normal Weight',
       'Overweight_Level_I': 'Overweight I',
       'Overweight_Level_II': 'Overweight II',
       'Obesity_Type_I': 'Obesity I',
       'Obesity_Type_II': 'Obesity II',
       'Obesity_Type_III': 'Obesity III'
      }

data['NObeyesdad_new'] = data['NObeyesdad'].map(obe).astype('category')
```

Algorithm 1: Correction on names of possible values (Obesity levels)

```
continuous_features = ["Age",
                       "Height",
                       "Weight",
                       "NCP",
                       "FCVC",
                       "CH2O",
                       "FAF",
```

```
                        "TUE"]
binary_features = ["family_history_with_overweight",
                    "FAVC",
                    "SMOKE",
                    "SCC"]
categorical_features = ["Gender",
                        "CAEC",
                        "CALC",
                        "MTRANS",
                        "NObeyesdad_new"]
```

Algorithm 2: Organization of features based on its data type.

```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy.stats import gaussian_kde, f_oneway, sem,
chi2_contingency, fisher_exact, chi2, ks_2samp
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.multitest import multipletests
from scipy.integrate import quad
from sklearn.metrics import roc_curve, auc, confusion_matrix
from statsmodels.stats.proportion import proportion_confint
```

Algorithm 3: Python libraries used for statistical development.

10.1 Task 1 Annexes

Feature	Type	Value	PMF	CDF
family_history_with_overweight	binary	yes	0.8176219801	0.8176219801
family_history_with_overweight	binary	no	0.1823780199	1
FAVC	binary	yes	0.8839412601	0.8839412601
FAVC	binary	no	0.1160587399	1
SMOKE	binary	no	0.9791567977	0.9791567977
SMOKE	binary	yes	0.02084320227	1
SCC	binary	no	0.9545239223	0.9545239223
SCC	binary	yes	0.04547607769	1
Gender	categorical	Male	0.5059213643	0.5059213643
Gender	categorical	Female	0.4940786357	1
CAEC	categorical	Sometimes	0.8360966367	0.8360966367
CAEC	categorical	Frequently	0.1146376125	0.9507342492
CAEC	categorical	Always	0.02510658456	0.9758408337
CAEC	categorical	no	0.02415916627	1
CALC	categorical	Sometimes	0.6636665088	0.6636665088
CALC	categorical	no	0.3027001421	0.9663666509
CALC	categorical	Frequently	0.03315963998	0.9995262909
CALC	categorical	Always	0.0004737091426	1
MTRANS	categorical	Public_Transportation	0.7484604453	0.7484604453
MTRANS	categorical	Automobile	0.2164850782	0.9649455234
MTRANS	categorical	Walking	0.02652771198	0.9914732354
MTRANS	categorical	Motorbike	0.005210800568	0.996684036
MTRANS	categorical	Bike	0.003315963998	1
NObeyesdad	categorical	Obesity I	0.166271909	0.166271909
NObeyesdad	categorical	Obesity III	0.1534817622	0.3197536712
NObeyesdad	categorical	Obesity II	0.1406916153	0.4604452866
NObeyesdad	categorical	Overweight I	0.1373756514	0.5978209379
NObeyesdad	categorical	Overweight II	0.1373756514	0.7351965893
NObeyesdad	categorical	Normal Weight	0.1359545239	0.8711511132
NObeyesdad	categorical	Insufficient Weight	0.1288488868	1

Table 3: PMF and CDF values for categorical and binary features.

```
def bootstrap_ci_values(data, statfunc=np.mean, n_boot=1000, ci=95):
    data = np.array(data)
    n = len(data)
    boots = []
    for _ in range(n_boot):
        s = np.random.choice(data, size=n, replace=True)
        boots.append(statfunc(s))
    lo = np.percentile(boots, (100-ci)/2)
    hi = np.percentile(boots, 100-(100-ci)/2)
    return lo, hi, boots
```

Algorithm 4: Bootstrap Confidence Intervals.

```
def pmf_with_bootstrap(series, n_boot=1000):
    vals = series.dropna()
    counts = vals.value_counts().sort_index()
    pmf = (counts / counts.sum()).astype(float)
    rows = []
    for cat in pmf.index:
        binary = (vals == cat).astype(int).values
        lo, hi, _ = bootstrap_ci_values(binary, statfunc=np.mean, n_boot=n_boot)
        rows.append({"category": cat,
                     "pmf": pmf[cat],
                     "ci_low": lo,
                     "ci_high": hi,
                     "count": int(counts[cat])})
    df_pmf = pd.DataFrame(rows).sort_values("pmf", ascending=False)
    return df_pmf
```

Algorithm 5: PMF of Discrete Values

```
def kde_pdf_cdf_with_boot(series, grid_n=500, boot_n=500):
    data = series.dropna().values
    if len(data) < 3:
        raise ValueError("Not enough data for KDE")
    grid = np.linspace(data.min(), data.max(), grid_n)
    kde = gaussian_kde(data)
    pdf = kde(grid)
    from numpy import trapz
    pdf = pdf / trapz(pdf, grid)
    cdf = np.cumsum(pdf)
    cdf = cdf / cdf[-1]
```

```
# bootstrap
    pdf_boot = np.empty((boot_n, grid_n))
    cdf_boot = np.empty((boot_n, grid_n))
    for i in range(boot_n):
        samp = np.random.choice(data, size=len(data), replace=True)
        kv = gaussian_kde(samp)
        p = kv(grid)
        p = p / trapz(p, grid)
        pdf_boot[i,:] = p
        c = np.cumsum(p); c = c / c[-1]
        cdf_boot[i,:] = c
    pdf_low = np.percentile(pdf_boot, 2.5, axis=0)
    pdf_high = np.percentile(pdf_boot, 97.5, axis=0)
    cdf_low = np.percentile(cdf_boot, 2.5, axis=0)
    cdf_high = np.percentile(cdf_boot, 97.5, axis=0)
    return grid, pdf, pdf_low, pdf_high, cdf, cdf_low, cdf_high
```

Algorithm 6: PDF and CDF for Continuous Features.

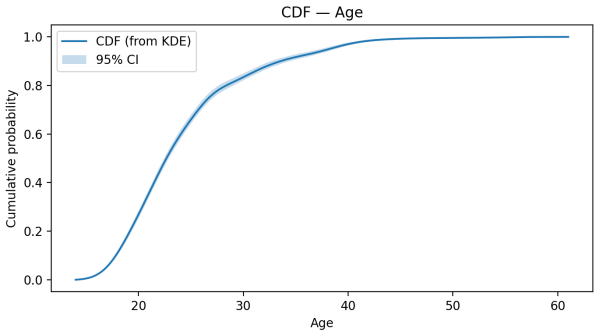

Figure 1: Age CFD.

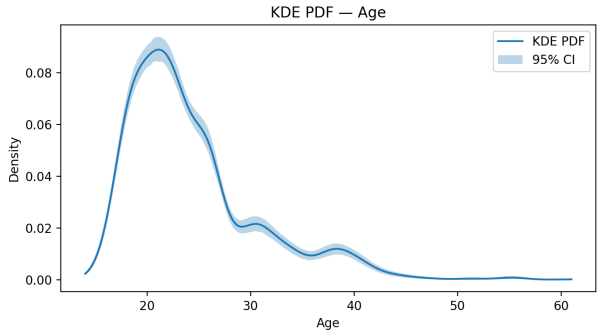

Figure 2: Age PDF.

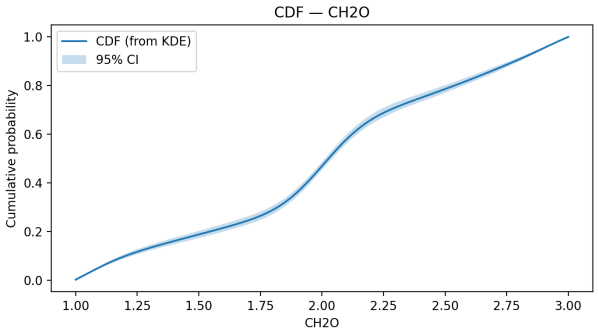

Figure 3: CH2O CDF.

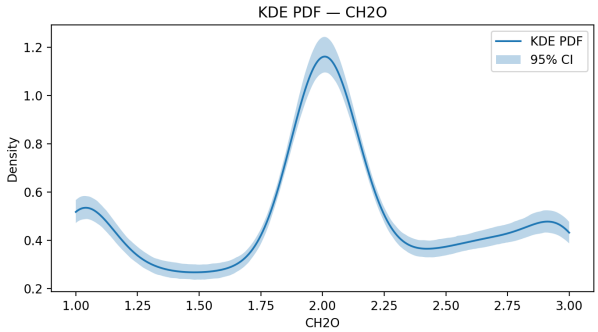

Figure 4: CH2O PDF.

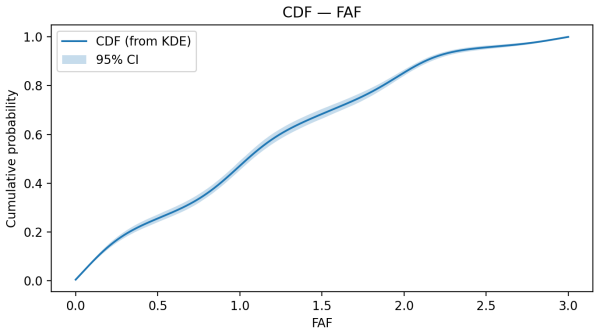

Figure 5: FAF CDF.

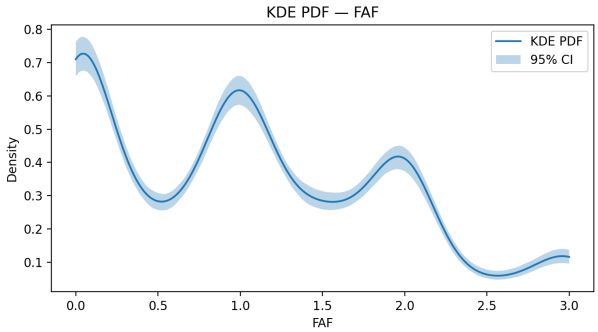

Figure 6: FAF PDF.

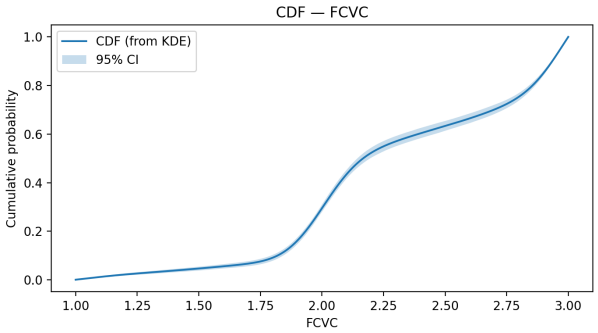

Figure 7: FCVC CDF.

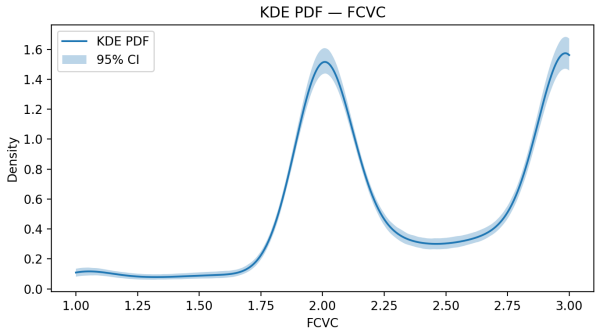

Figure 8: FCVC PDF.

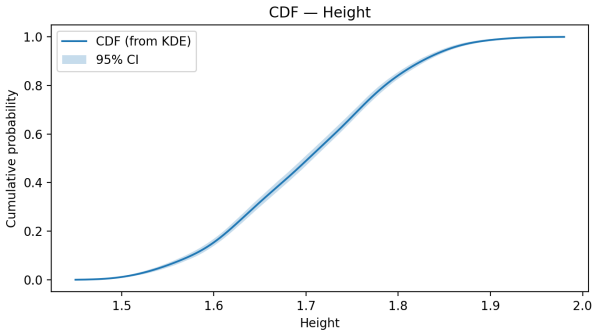

Figure 9: Height CDF.

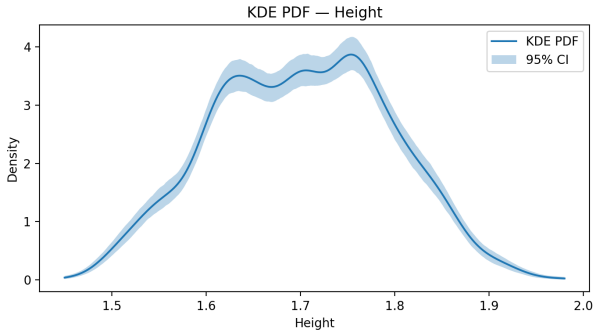

Figure 10: Height PDF.

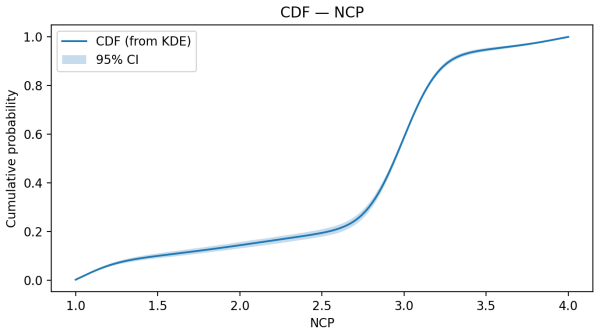

Figure 11: NCP CDF.

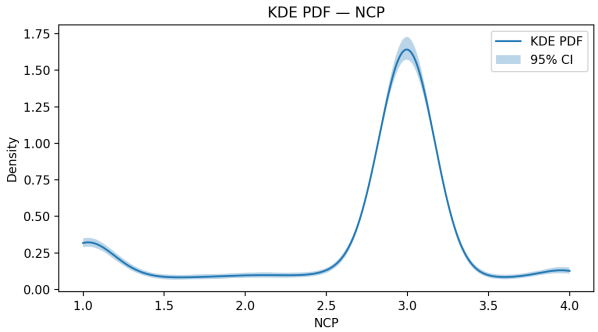

Figure 12: NCP PDF.

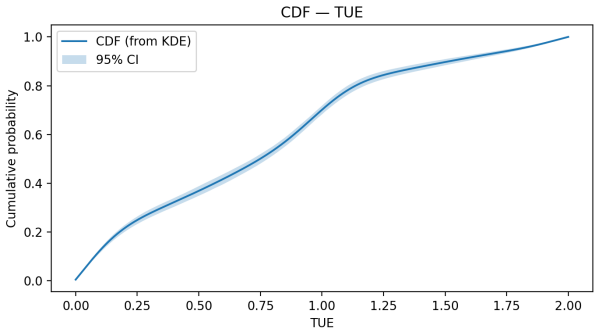

Figure 13: TUE CDF.

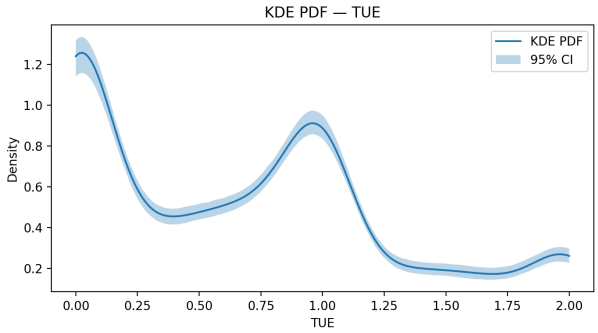

Figure 14: TUE PDF.

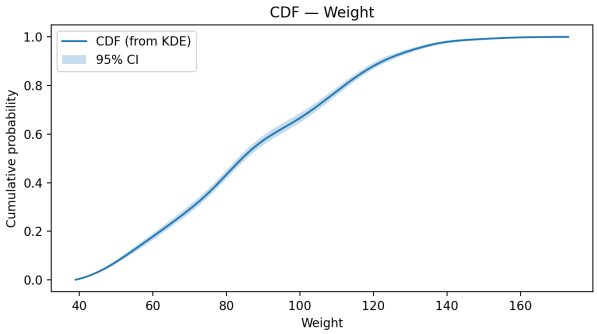

Figure 15: Weight CDF.

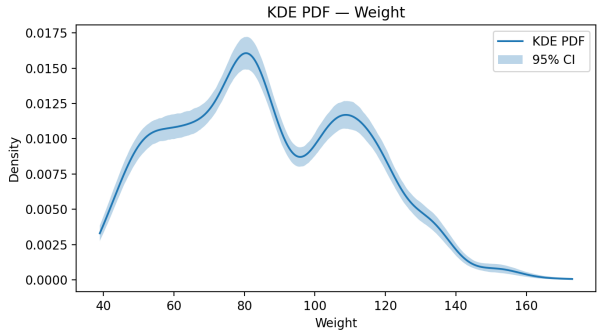

Figure 16: Weight PDF.

10.2 Task 2 Annexes

```
def bootstrap_ci(data, statfunc, n_boot=2000, ci=95):
    data = np.array(data.dropna())
    if len(data) == 0:
        return np.nan, np.nan, np.nan, []
    boots = []
    n = len(data)
    for _ in range(n_boot):
        s = np.random.choice(data, size=n, replace=True)
        boots.append(statfunc(s))
    boots = np.array(boots)
    lo = np.percentile(boots, (100-ci)/2)
    hi = np.percentile(boots, 100-(100-ci)/2)
    est = statfunc(data)
    return est, lo, hi, boots
```

Algorithm 7: PDF and CDF for Continuous Features.

```
def welch_ttest(group1, group2):
    x1 = np.array(group1.dropna())
    x2 = np.array(group2.dropna())
    n1 = len(x1); n2 = len(x2)
    if n1 < 2 or n2 < 2:
        return np.nan, np.nan, np.nan
    m1 = x1.mean(); m2 = x2.mean()
    s1 = x1.var(ddof=1); s2 = x2.var(ddof=1)
    se = np.sqrt(s1/n1 + s2/n2)
```

```
    t = (m1 - m2) / se
    num = (s1/n1 + s2/n2)**2
    den = ( (s1**2) / ( (n1**2) * (n1 - 1) ) ) +
    ( (s2**2) / ( (n2**2) * (n2 - 1) ) )
    df = num / den if den != 0 else np.nan
    p = stats.t.sf(np.abs(t), df)*2
    return t, df, p
```

Algorithm 8: PDF and CDF for Continuous Features.

```
    formula = f"{feat} ~ C(NObeyesdad_new)"
    model = smf.ols(formula, data=sub).fit()
    anova_table = sm.stats.anova_lm(model, typ=2)
    F = anova_table.loc['C(NObeyesdad_new)'
    ,'F'] if 'C(NObeyesdad_new)' in anova_table.index else np.nan
    pval = anova_table.loc['C(NObeyesdad_new)'
    ,'PR(>F)'] if 'C(NObeyesdad_new)' in anova_table.index else np.nan

    # 2) aov() equivalent: scipy f_oneway across groups
    groups = [g[feat].values for n,g in sub.groupby('NObeyesdad_new')]
    try:
        f_sc, p_sc = stats.f_oneway(*groups)
    except Exception:
    f_sc, p_sc = np.nan, np.nan

anova_summary_rows.append({'feature':feat,'F_lm':float(F),
                           'p_lm':float(pval),'F_foneway':float(f_sc),
                           'p_foneway':float(p_sc)})
```

Algorithm 9: Use of OLS model to implement One-way ANOVA.

Feature	Level	n	Mean	Variance
Age	Insufficient Weight	272	19.7832	7.1263
Age	Normal Weight	287	21.7387	25.9769
Age	Obesity	972	25.8062	35.0343
Age	Overweight	580	25.2073	54.3685
Height	Insufficient Weight	272	1.6911	0.00994
Height	Normal Weight	287	1.6766	0.00894
Height	Obesity	972	1.7156	0.00792
Height	Overweight	580	1.6958	0.00866
Weight	Insufficient Weight	272	49.9063	36.1286
Weight	Normal Weight	287	62.1551	86.4162
Weight	Obesity	972	109.0823	301.2134
Weight	Overweight	580	78.1760	86.7723
NCP	Insufficient Weight	272	2.9144	0.8123
NCP	Normal Weight	287	2.7387	0.7601
NCP	Obesity	972	2.7168	0.3831
NCP	Overweight	580	2.4999	0.7457
FCVC	Insufficient Weight	272	2.4808	0.3420
FCVC	Normal Weight	287	2.3345	0.3493
FCVC	Obesity	972	2.5201	0.2627
FCVC	Overweight	580	2.2626	0.2188
CH2O	Insufficient Weight	272	1.8713	0.3628
CH2O	Normal Weight	287	1.8502	0.4075
CH2O	Obesity	972	2.0726	0.3735
CH2O	Overweight	580	2.0419	0.3424
FAF	Insufficient Weight	272	1.2501	0.7338
FAF	Normal Weight	287	1.2474	1.0330
FAF	Obesity	972	0.8749	0.5923
FAF	Overweight	580	1.0074	0.7039
TUE	Insufficient Weight	272	0.8395	0.4137
TUE	Normal Weight	287	0.6760	0.4716
TUE	Obesity	972	0.6033	0.2982
TUE	Overweight	580	0.6551	0.4039

Table 4: Conditional Means and Variances by Obesity Level.

Feature	F Statistic	p-value
Age	94.8926	1.32×10^{-57}
Height	16.3647	1.63×10^{-10}
Weight	1992.5183	0
NCP	20.3693	5.19×10^{-13}
FCVC	33.2694	5.25×10^{-21}
CH2O	15.3559	6.96×10^{-10}
FAF	23.5881	5.15×10^{-15}
TUE	10.9011	4.19×10^{-07}

Table 5: One-way ANOVA Results for Differences Across Obesity Levels.

Feature	F (LM)	p (LM)	F (ANOVA)	p (ANOVA)
Age	94.8926	1.32×10^{-57}	94.8926	1.32×10^{-57}
Height	16.3647	1.63×10^{-10}	16.3647	1.63×10^{-10}
Weight	1992.5183	0	1992.5183	0
NCP	20.3693	5.19×10^{-13}	20.3693	5.19×10^{-13}
FCVC	33.2694	5.25×10^{-21}	33.2694	5.25×10^{-21}
CH2O	15.3559	6.96×10^{-10}	15.3559	6.96×10^{-10}
FAF	23.5881	5.15×10^{-15}	23.5881	5.15×10^{-15}
TUE	10.9011	4.19×10^{-7}	10.9011	4.19×10^{-7}

Table 6: Linear model and One-Way ANOVA results for predictive features.

Feature	Shapiro Statistic	Shapiro p	Levene Statistic	Levene p
Age	0.8806	1.58×10^{-37}	44.1969	1.07×10^{-27}
Height	0.9926	7.86×10^{-9}	1.3070	0.2704
Weight	0.9784	2.34×10^{-17}	123.0533	1.93×10^{-73}
NCP	0.7940	1.09×10^{-45}	34.6433	7.50×10^{-22}
FCVC	0.9163	1.14×10^{-32}	9.9703	1.59×10^{-6}
CH2O	0.9595	8.02×10^{-24}	5.8795	5.40×10^{-4}
FAF	0.9546	4.21×10^{-25}	10.4891	7.56×10^{-7}
TUE	0.9177	1.90×10^{-32}	14.7883	1.57×10^{-9}

Table 7: Shapiro–Wilk Normality and Levene Homogeneity tests across features.

Feature	Group 1	Group 2	Mean Diff	p-adj	Lower	Upper	Reject
Age							
	Insufficient Weight	Normal Weight	1.9554	0.0006	0.6586	3.2523	True
	Insufficient Weight	Obesity	6.0229	0.0000	4.9717	7.0742	True
	Insufficient Weight	Overweight	5.4241	0.0000	4.2979	6.5503	True
	Normal Weight	Obesity	4.0675	0.0000	3.0380	5.0970	True
	Normal Weight	Overweight	3.4687	0.0000	2.3626	4.5747	True
	Obesity	Overweight	-0.5989	0.2220	-1.4029	0.2052	False
Height							
	Insufficient Weight	Normal Weight	-0.0145	0.2456	-0.0346	0.0056	False
	Insufficient Weight	Obesity	0.0244	0.0007	0.0082	0.0407	True
	Insufficient Weight	Overweight	0.0047	0.9013	-0.0128	0.0221	False
	Normal Weight	Obesity	0.0390	0.0000	0.0230	0.0549	True
	Normal Weight	Overweight	0.0192	0.0207	0.0021	0.0363	True
	Obesity	Overweight	-0.0198	0.0003	-0.0322	-0.0073	True
Weight							
	Insufficient Weight	Normal Weight	12.2487	0.0000	9.3376	15.1599	True
	Insufficient Weight	Obesity	59.1760	0.0000	56.8162	61.5358	True
	Insufficient Weight	Overweight	28.2697	0.0000	25.7416	30.7979	True
	Normal Weight	Obesity	46.9273	0.0000	44.6162	49.2384	True
	Normal Weight	Overweight	16.0210	0.0000	13.5382	18.5038	True
	Obesity	Overweight	-30.9063	0.0000	-32.7113	-29.1013	True
FAF							
	Insufficient Weight	Normal Weight	-0.0027	1.0000	-0.1849	0.1794	False
	Insufficient Weight	Obesity	-0.3752	0.0000	-0.5229	-0.2276	True
	Insufficient Weight	Overweight	-0.2427	0.0005	-0.4009	-0.0845	True
	Normal Weight	Obesity	-0.3725	0.0000	-0.5171	-0.2279	True
	Normal Weight	Overweight	-0.2400	0.0004	-0.3953	-0.0846	True
	Obesity	Overweight	0.1325	0.0137	0.0196	0.2455	True
TUE							
	Insufficient Weight	Normal Weight	-0.1635	0.0077	-0.2951	-0.0319	True
	Insufficient Weight	Obesity	-0.2361	0.0000	-0.3428	-0.1295	True
	Insufficient Weight	Overweight	-0.1843	0.0002	-0.2986	-0.0701	True
	Normal Weight	Obesity	-0.0726	0.2795	-0.1771	0.0318	False
	Normal Weight	Overweight	-0.0208	0.9641	-0.1330	0.0914	False
	Obesity	Overweight	0.0518	0.3605	-0.0298	0.1334	False

Table 8: Combined Tukey HSD Post-Hoc Comparisons Across Obesity Levels

10.3 Task 3 Annexes

```
def run_stat_test(contingency, is_binary):

    chi2 = np.nan
    dof = np.nan
    p = np.nan
    min_expected = np.nan
    test_used = "N/A"

    if contingency.size == 0 or
    contingency.shape[0] < 2 or
    contingency.shape[1] < 2:
    return chi2, dof, p, min_expected, test_used

    try:
        chi2, p, dof, expected = chi2_contingency(contingency)
        min_expected = expected.min()
        test_used = "Chi-square"
    except:
        if contingency.shape == (2, 2):
            try:
                _, p = fisher_exact(contingency)
                test_used = "Fisher"
            except:
                pass
        return chi2, dof, p, min_expected, test_used
```

Algorithm 10: Helper function to run Chi-square

Feature	Variable Type	Chi-Squared	df
family_history_with_overweight	Binary	575.571169	3
FAVC	Binary	186.518899	3
SMOKE	Binary	13.901526	3
SCC	Binary	79.642730	3
Gender	Categorical	32.196306	3
CAEC	Categorical	715.051862	9
CALC	Categorical	90.328875	9
MTRANS	Categorical	150.150517	12
NObeyesdad_new	Categorical	6333.000000	9

Table 9: Chi-square test results (Part 1).

P-value	Min Expected	Test	FDR P-value	Reject H_0
$1.99e^{-124}$	49.606821	Chi-square	$5.97e^{-124}$	True
$3.45e^{-40}$	31.567977	Chi-square	$7.76e^{-40}$	True
$3.04e^{-03}$	5.669351	Chi-square	$3.04e^{-03}$	True
$3.66e^{-17}$	12.369493	Chi-square	$5.49e^{-17}$	True
$4.76e^{-07}$	134.389389	Chi-square	$5.35e^{-07}$	True
$4.01e^{-148}$	6.571293	Chi-square	$1.81e^{-147}$	True
$1.40e^{-15}$	0.128849	Chi-square	$1.80e^{-15}$	True
$5.28e^{-26}$	0.901942	Chi-square	$9.51e^{-26}$	True
$0.00e^{+00}$	35.046897	Chi-square	$0.00e^{+00}$	True

Table 10: Chi-square test results (Part 2).

10.4 Task 4 Annexes

```
def compute_pmf(series):
    counts = series.value_counts().sort_index()
    pmf = counts / counts.sum()
    cdf = pmf.cumsum()
    return pd.DataFrame({"category": pmf.index,
    "pmf": pmf.values,
    "cdf": cdf.values})
```

Algorithm 11: Utility function for PMF computation.

```
for col in tqdm(binary_features + categorical_features):

    results_dict = {}

    for lvl in levels:
        subset = df[df[target] == lvl][col]
        pmf_df = compute_pmf(subset)
        results_dict[lvl] = pmf_df

    for _, row in pmf_df.iterrows():
        pmf_summary_rows.append({
            "Feature": col,
            "Category": row["category"],
            "Obesity Level": lvl,
            "PMF": row["pmf"]
        })
        cdf_summary_rows.append({
```

```
        "Feature": col,
        "Category": row["category"],
        "Obesity Level": lvl,
        "CDF": row["cdf"]
    })
```

Algorithm 12: Process for discrete features (PMF + CDF)

Feature	Category	I. Weight	N. Weight	Obesity	O. Weight
CAEC	Always	0.00735	0.12195	0.00823	0.01379
CAEC	Frequently	0.44485	0.28920	0.00823	0.05172
CAEC	Sometimes	0.53676	0.55401	0.98148	0.87241
CAEC	no	0.01103	0.03484	0.00206	0.06207
CALC	Always	–	0.00348	–	–
CALC	Frequently	0.00368	0.06272	0.01646	0.06034
CALC	Sometimes	0.56618	0.56098	0.73971	0.63276
CALC	no	0.43015	0.37282	0.24383	0.30690
FAVC	no	0.18750	0.27526	0.01955	0.16552
FAVC	yes	0.81250	0.72474	0.98045	0.83448
Gender	Female	0.63603	0.49129	0.49486	0.42759
Gender	Male	0.36397	0.50871	0.50514	0.57241
MTRANS	Automobile	0.16912	0.15679	0.21193	0.27586
MTRANS	Bike	–	0.01394	0.00103	0.00345
MTRANS	Motorbike	–	0.02091	0.00309	0.00345
MTRANS	Public Transportation	0.80882	0.69686	0.78086	0.69138
MTRANS	Walking	0.02206	0.11150	0.00309	0.02586
SCC	no	0.91912	0.89547	0.99691	0.92931
SCC	yes	0.08088	0.10453	0.00309	0.07069
SMOKE	no	0.99632	0.95470	0.97737	0.98621
SMOKE	yes	0.00368	0.04530	0.02263	0.01379
family_history_with_overweight	no	0.53676	0.45993	0.00823	0.17069
family_history_with_overweight	yes	0.46324	0.54007	0.99177	0.82931

Table 11: Category proportions by weight status (PMF).

Feature	Category	I. Weight	N. Weight	Obesity	O. Weight
CAEC	Always	0.00735	0.12195	0.00823	0.01379
CAEC	Frequently	0.45221	0.41115	0.01646	0.06552
CAEC	Sometimes	0.98897	0.96516	0.99794	0.93793
CAEC	no	1	1	1	1
CALC	Always	–	0.00348	–	–
CALC	Frequently	0.00368	0.06620	0.01646	0.06034
CALC	Sometimes	0.56985	0.62718	0.75617	0.69310
CALC	no	1	1	1	1
FAVC	no	0.18750	0.27526	0.01955	0.16552
FAVC	yes	1	1	1	1
Gender	Female	0.63603	0.49129	0.49486	0.42759
Gender	Male	1	1	1	1
MTRANS	Automobile	0.16912	0.15679	0.21193	0.27586
MTRANS	Bike	–	0.17073	0.21296	0.27931
MTRANS	Motorbike	–	0.19164	0.21605	0.28276
MTRANS	Public Transportation	0.97794	0.88850	0.99691	0.97414
MTRANS	Walking	1	1	1	1
SCC	no	0.91912	0.89547	0.99691	0.92931
SCC	yes	1	1	1	1
SMOKE	no	0.99632	0.95470	0.97737	0.98621
SMOKE	yes	1	1	1	1
family_history_with_overweight	no	0.53676	0.45993	0.00823	0.17069
family_history_with_overweight	yes	1	1	1	1

Table 12: Category proportions by weight status (Table 2).

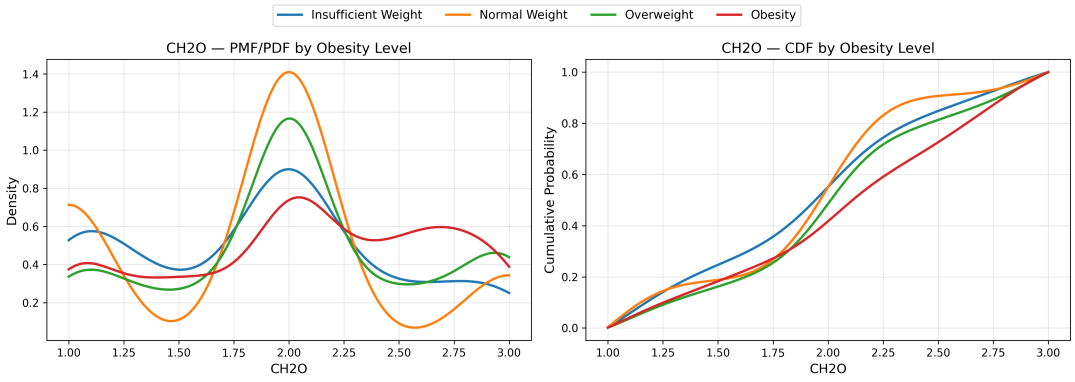

Figure 17: KDEs of CH2O stratified by obesity level.

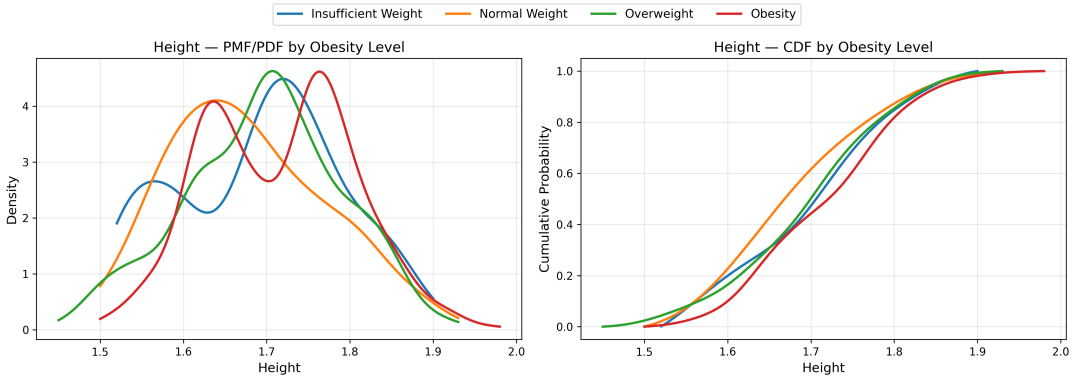

Figure 18: KDEs of Height stratified by obesity level.

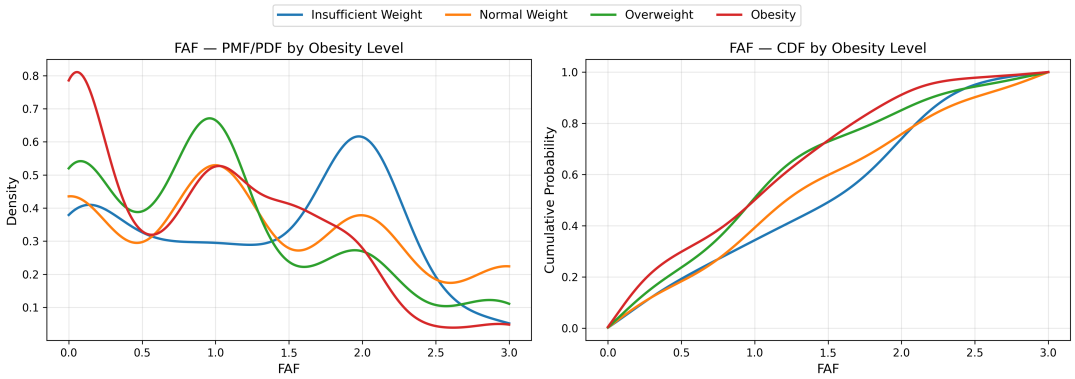

Figure 19: KDEs of FAF stratified by obesity level.

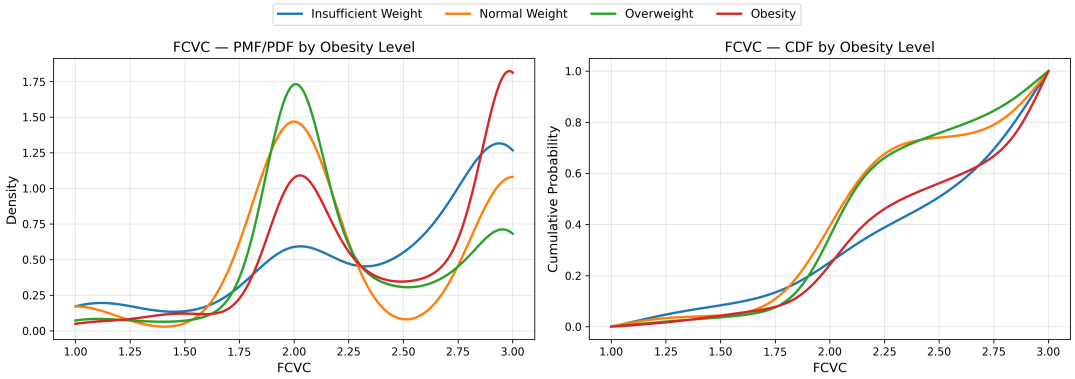

Figure 20: KDEs of FCVC stratified by obesity level.

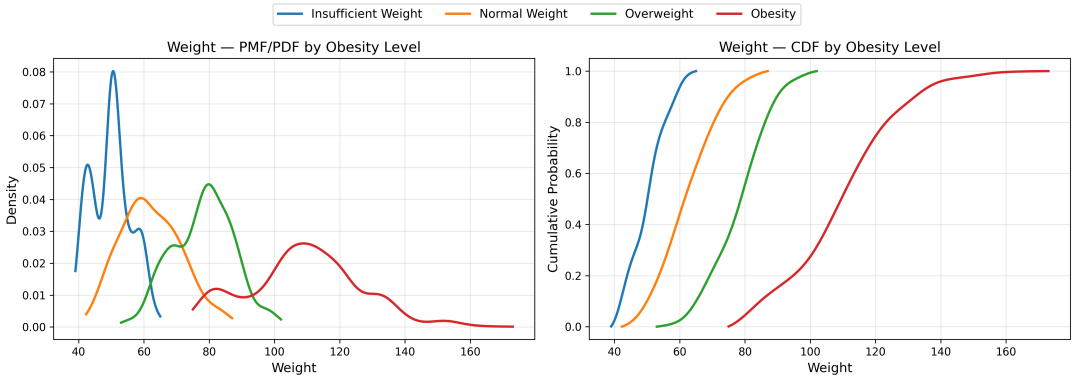

Figure 21: KDEs of Weight stratified by obesity level.

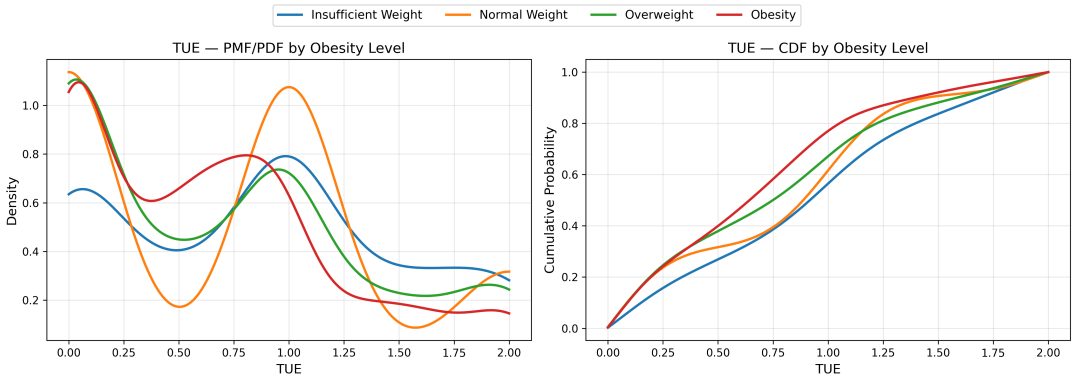

Figure 22: KDEs of TUE stratified by obesity level.

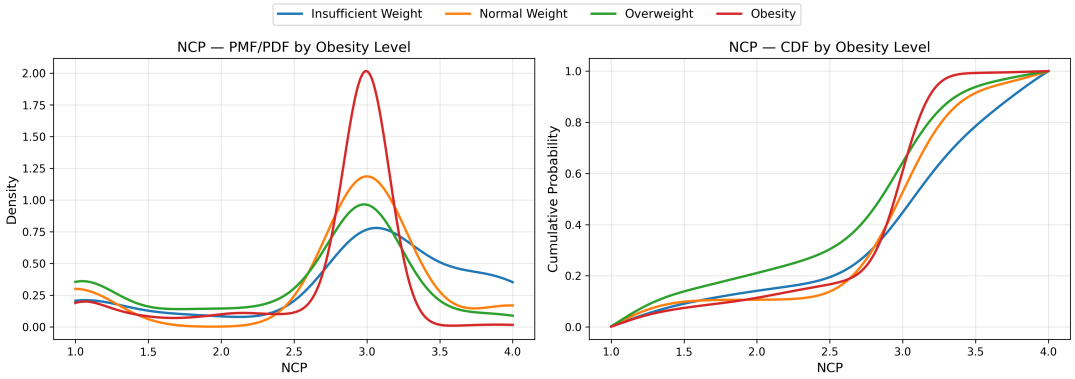

Figure 23: KDEs of NCP stratified by obesity level.

10.5 Task 5 Annexes

```
def kde_model_based_mean_variance(series, grid_points=2000):
    x = series.dropna().values
    if len(x) < 5:
        return np.nan, np.nan

    kde = gaussian_kde(x)

    x_grid = np.linspace(x.min(), x.max(), grid_points)
    pdf_vals = kde(x_grid)

    pdf_vals /= np.trapz(pdf_vals, x_grid)

    mean_model = np.trapz(x_grid * pdf_vals, x_grid)
    var_model = np.trapz((x_grid - mean_model)**2 * pdf_vals, x_grid)
    return mean_model, var_model
```

Algorithm 13: KDE for model-based (mean and variance.)

Feature	Level	Mean (Model)	Mean (Sample)	Equivalent?
Age	Insufficient Weight	19.8516	19.7832	False
Age	Normal Weight	21.7320	21.7387	False
Age	Obesity	25.8228	25.8062	False
Age	Overweight	25.4628	25.2073	False
Height	Insufficient Weight	1.6990	1.6911	False
Height	Normal Weight	1.6794	1.6766	False
Height	Obesity	1.7160	1.7156	True
Height	Overweight	1.6959	1.6958	True
Weight	Insufficient Weight	50.0973	49.9063	False
Weight	Normal Weight	62.1590	62.1551	False
Weight	Obesity	109.6744	109.0823	False
Weight	Overweight	78.1365	78.1761	False
NCP	Insufficient Weight	2.9261	2.9144	False
NCP	Normal Weight	2.8581	2.7387	False
NCP	Obesity	2.7720	2.7168	False
NCP	Overweight	2.6301	2.4999	False
FCVC	Insufficient Weight	2.3548	2.4808	False
FCVC	Normal Weight	2.1842	2.3345	False
FCVC	Obesity	2.3675	2.5201	False
FCVC	Overweight	2.1926	2.2626	False
CH2O	Insufficient Weight	1.9222	1.8713	False
CH2O	Normal Weight	1.9203	1.8502	False
CH2O	Obesity	2.0871	2.0726	False
CH2O	Overweight	2.0233	2.0419	False
FAF	Insufficient Weight	1.3871	1.2501	False
FAF	Normal Weight	1.3316	1.2474	False
FAF	Obesity	1.0256	0.8749	False
FAF	Overweight	1.1096	1.0074	False
TUE	Insufficient Weight	0.9080	0.8395	False
TUE	Normal Weight	0.8128	0.6760	False
TUE	Obesity	0.6860	0.6033	False
TUE	Overweight	0.7663	0.6551	False

Table 13: Model-based vs. sample means across obesity levels.

10.6 Task 6 Annexes

```
def map_target_to_4(df):
    if target_new in df.columns:
        return df
    map4 = {
    'Insufficient_Weight': 'Insufficient Weight',
    'Normal_Weight': 'Normal Weight',
    'Overweight_Level_I': 'Overweight',
    'Overweight_Level_II': 'Overweight',
    'Obesity_Type_I': 'Obesity',
    'Obesity_Type_II': 'Obesity',
    'Obesity_Type_III': 'Obesity'
    }
    df[target_new] = df[target_orig].map(map4)
    df[target_new] = pd.Categorical(df[target_new],
    categories=levels_order,
    ordered=True)
    return df

def kde_fit_and_grid(data, grid):
    x = np.asarray(data.dropna())
    if len(x) < 3:
        return np.zeros_like(grid)
    kde = gaussian_kde(x)
    pdf = kde(grid)
    pdf = pdf / np.trapz(pdf, grid)
    return pdf

def bootstrap_kde_bands(data, grid, n_boot=N_BOOT_KDE):
    x = np.asarray(data.dropna())
    if len(x) < 3:
        return np.zeros_like(grid), np.zeros_like(grid), np.zeros_like(grid)
    pdfs = np.empty((n_boot, len(grid)))
    n = len(x)
    for i in range(n_boot):
        samp = np.random.choice(x, size=n, replace=True)
        try:
            pdfs[i,:] = gaussian_kde(samp)(grid)
        except Exception:
            pdfs[i,:] = np.zeros_like(grid)
```

```
    for i in range(n_boot):
        arr = pdfs[i,:]
        s = np.trapz(arr, grid)
        if s > 0:
            pdfs[i,:] = arr / s
    pdf_mean = np.mean(pdfs, axis=0)
    pdf_low = np.percentile(pdfs, 2.5, axis=0)
    pdf_high = np.percentile(pdfs, 97.5, axis=0)
    return pdf_mean, pdf_low, pdf_high

def find_discriminative_intervals(grid,
    pdf_low_g,
    pdf_high_g,
    other_pdfs_low,
    other_pdfs_high,
    min_width=0.01):
    other_high_max = np.maximum.reduce(other_pdfs_high) if len(other_pdfs_high)>0
    else np.full_like(grid, -np.inf)
    other_low_min = np.minimum.reduce(other_pdfs_low) if len(other_pdfs_low)>0
    else np.full_like(grid, np.inf)

    cond_higher = pdf_low_g > other_high_max
    cond_lower = pdf_high_g < other_low_min

def grid_to_intervals(cond_mask):
    intervals = []
    i = 0
    while i < len(cond_mask):
        if cond_mask[i]:
            j = i
            while j+1 < len(cond_mask) and cond_mask[j+1]:
                j += 1
            a = grid[i]
            b = grid[j]
            if (b - a) >= min_width*(grid.max()-grid.min()):
                intervals.append((a, b))
            i = j+1
        else:
            i += 1
    return intervals
```

```
    higher_ints = grid_to_intervals(cond_higher)
    lower_ints = grid_to_intervals(cond_lower)
    return higher_ints, lower_ints

    def youden_threshold(y_true, scores):
        fpr, tpr, thr = roc_curve(y_true, scores)
        J = tpr - fpr
        idx = np.argmax(J)
        return thr[idx], fpr[idx], 1 - (1 - tpr[idx])

    def bootstrap_auc_ci(y_true, scores, n_boot=N_BOOT_ROC, seed=42):
        rng = np.random.RandomState(seed)
        n = len(y_true)
        aucs = []
        for i in range(n_boot):
            idx = rng.choice(np.arange(n), size=n, replace=True)
            if len(np.unique(y_true[idx])) < 2:
            aucs.append(np.nan)
            else:
                aucs.append(roc_auc_score(y_true[idx], scores[idx]))
        aucs = np.array(aucs)
        lo = np.nanpercentile(aucs, 2.5)
        hi = np.nanpercentile(aucs, 97.5)
        return np.nanmean(aucs), lo, hi, aucs

    def bootstrap_rates_at_threshold(y_true,
        scores, threshold,
        n_boot=N_BOOT_ROC, seed=42):
        rng = np.random.RandomState(seed)
        n = len(y_true)
        fprs = []
        fnrs = []
        for i in range(n_boot):
            idx = rng.choice(np.arange(n), size=n, replace=True)
            yt = y_true[idx]
            sc = scores[idx]
            yhat = (sc >= threshold).astype(int)
            tn, fp, fn, tp = confusion_matrix(yt, yhat, labels=[0,1]).ravel()
            fpr = fp / (fp + tn) if (fp + tn) > 0 else np.nan
            fnr = fn / (fn + tp) if (fn + tp) > 0 else np.nan
            fprs.append(fpr)
```

```
        fnrs.append(fnr)
    return np.nanpercentile(fprs, 2.5), np.nanpercentile(fprs,97.5),
    np.nanpercentile(fnrs,2.5),
    np.nanpercentile(fnrs,97.5),
    np.nanmean(fprs),
    np.nanmean(fnrs)
```

Algorithm 14: Complete functions used for the 4 parts of Task 6.

```
def encode_to_scores(series):
if series.name in ordinal_mappings:
    return series.
    map(ordinal_mappings[series.name]).
    astype(float).
    fillna(np.nan).
    values
else:
    if series.dtype == 'object' or str(series.dtype).startswith('category'):
        cats = sorted(series.dropna().unique().tolist(), key=lambda x: str(x))
        lookup = {c:i for i,c in enumerate(cats)}
        return series.map(lookup).astype(float).fillna(np.nan).values
    else:
        return series.astype(float).fillna(np.nan).values

def bootstrap_metric_ci(y_true,
                        scores,
                        threshold,
                        metric_func,
                        n_boot=1000,
                        seed=42):
    rng = np.random.RandomState(seed)
    n = len(y_true)
    boots = []
    for i in range(n_boot):
        idx = rng.choice(np.arange(n), size=n, replace=True)
        yt = y_true[idx]
        sc = scores[idx]
        yhat = (sc >= threshold).astype(int)
        boots.append(metric_func(yt, yhat))
    boots = np.array(boots)
```

```
return np.nanmean(boots),
np.nanpercentile(boots,2.5),
np.nanpercentile(boots,97.5)
```

Algorithm 15: One-vs-rest implementation (functions, including bootstrap).

```
yhat_full = (sc_valid >= best_thr).astype(int)
tn, fp, fn, tp = confusion_matrix(y_valid, yhat_full, labels=[0,1]).ravel()
accuracy = accuracy_score(y_valid, yhat_full)
sensitivity = recall_score(y_valid, yhat_full, zero_division=0)  # TPR
specificity = tn / (tn+fp) if (tn+fp)>0 else np.nan
precision = precision_score(y_valid, yhat_full, zero_division=0)
f1 = f1_score(y_valid, yhat_full, zero_division=0)
# cohens kappa
kappa = cohen_kappa_score(y_valid, yhat_full)
```

Algorithm 16: Computation of metrics.

Feature	Obesity Level	Discriminative Intervals (a,b]
Age	Insufficient Weight	(15.929, 21.104]
Age	Normal Weight	(14.000, 23.880]
Age	Overweight	(28.161, 39.217]
Age	Obesity	(23.080, 32.254], (37.618, 41.099]
Height	Insufficient Weight	(1.481, 1.583], (1.692, 1.755], (1.863, 1.877]
Height	Normal Weight	(1.518, 1.683]
Height	Overweight	(1.457, 1.503], (1.663, 1.746], (1.828, 1.839]
Height	Obesity	(1.606, 1.662], (1.738, 1.849]
Weight	Insufficient Weight	(39.000, 60.461]
Weight	Normal Weight	(47.048, 47.450], (51.743, 75.350]
Weight	Overweight	(63.681, 94.666]
Weight	Obesity	(91.044, 140.808]
NCP	Insufficient Weight	(3.276, 4.000]
NCP	Normal Weight	(1.000, 1.087], (2.486, 2.844], (3.141, 3.517]
NCP	Overweight	(1.000, 2.742]
NCP	Obesity	(2.763, 3.207]
FCVC	Insufficient Weight	(1.124, 1.238], (2.361, 2.948]
FCVC	Normal Weight	(1.000, 1.080], (1.611, 2.251]
FCVC	Overweight	(1.773, 2.299]
FCVC	Obesity	(2.792, 3.000]
CH2O	Insufficient Weight	(1.000, 1.779], (2.393, 2.411]
CH2O	Normal Weight	(1.000, 1.158], (1.723, 2.243]
CH2O	Overweight	(1.711, 2.211], (2.766, 3.000]
CH2O	Obesity	(1.334, 1.607], (2.265, 3.000]
FAF	Insufficient Weight	(1.486, 2.613]
FAF	Normal Weight	(2.246, 3.000]
FAF	Overweight	(0.195, 1.288]
FAF	Obesity	(0.000, 0.408], (1.015, 1.706]
TUE	Insufficient Weight	(1.005, 2.000]
TUE	Normal Weight	(0.000, 0.026], (0.861, 1.303], (1.964, 2.000]
TUE	Overweight	(0.000, 0.513]
TUE	Obesity	(0.000, 0.889]

Table 14: Discriminative intervals $(a, b]$ where the conditional PDF for a given obesity level is statistically distinct from the remaining levels (Task 6).

Feature	Obesity Level	Type I Error	CI Lower	CI Upper
Age	Insufficient Weight	0.995119	0.991830	0.997838
	Normal Weight	0.000000	0.000000	0.000000
	Overweight	0.200450	0.182218	0.220134
	Obesity	0.368097	0.341528	0.394227
Height	Insufficient Weight	0.560169	0.538880	0.582382
	Normal Weight	0.027359	0.019737	0.035088
	Overweight	0.594287	0.570852	0.617913
	Obesity	0.331367	0.303775	0.356475
Weight	Insufficient Weight	0.000000	0.000000	0.000000
	Normal Weight	0.952980	0.943517	0.962719
	Overweight	0.733086	0.710647	0.755732
	Obesity	0.045948	0.033363	0.058824
NCP	Insufficient Weight	0.062639	0.051115	0.073953
	Normal Weight	0.109038	0.094846	0.124452
	Overweight	0.099294	0.084242	0.114304
	Obesity	0.859130	0.840211	0.879719
FCVC	Insufficient Weight	0.487165	0.464927	0.510060
	Normal Weight	0.000000	0.000000	0.000000
	Overweight	0.942637	0.930764	0.954278
	Obesity	0.512706	0.482880	0.539069
CH2O	Insufficient Weight	0.000000	0.000000	0.000000
	Normal Weight	0.468072	0.445162	0.491790
	Overweight	0.722017	0.696930	0.745918
	Obesity	0.311177	0.284460	0.337138
FAF	Insufficient Weight	0.194350	0.176726	0.212072
	Normal Weight	0.083561	0.070175	0.097039
	Overweight	0.691472	0.669481	0.715235
	Obesity	0.410076	0.379280	0.438982
TUE	Insufficient Weight	0.307986	0.286025	0.327896
	Normal Weight	0.201698	0.182566	0.220408
	Overweight	0.179291	0.160010	0.197926
	Obesity	0.690676	0.663740	0.716440

Table 15: Type I Error for False Positive Rate.

Feature	Obesity Level	Power	CI Lower	CI Upper
Age	Insufficient Weight	0.992559	0.981618	1.000000
	Normal Weight	0.000000	0.000000	0.000000
	Overweight	0.321112	0.286207	0.356940
	Obesity	0.710703	0.684156	0.737654
Weight	Insufficient Weight	0.000000	0.000000	0.000000
	Normal Weight	0.972223	0.951220	0.989547
	Overweight	0.979238	0.967241	0.989655
	Obesity	0.843824	0.822016	0.866255
NCP	Insufficient Weight	0.403599	0.345588	0.459559
	Normal Weight	0.101125	0.069686	0.135976
	Overweight	0.128747	0.101724	0.155172
	Obesity	0.959332	0.946502	0.971193
FCVC	Insufficient Weight	0.654831	0.591912	0.709559
	Normal Weight	0.000000	0.000000	0.000000
	Overweight	0.956900	0.939655	0.972414
	Obesity	0.728588	0.700617	0.755144
CH2O	Insufficient Weight	0.000000	0.000000	0.000000
	Normal Weight	0.139394	0.101045	0.181185
	Overweight	0.787912	0.753448	0.820690
	Obesity	0.553448	0.522634	0.585417
FAF	Insufficient Weight	0.411261	0.352941	0.466912
	Normal Weight	0.143024	0.101045	0.188153
	Overweight	0.745241	0.710345	0.779310
	Obesity	0.409619	0.379630	0.437269
TUE	Insufficient Weight	0.513147	0.452206	0.577206
	Normal Weight	0.126770	0.090592	0.167247
	Overweight	0.214929	0.181034	0.246552
	Obesity	0.788914	0.762346	0.813812
Height	Insufficient Weight	0.599140	0.536765	0.658088
	Normal Weight	0.023794	0.006969	0.041812
	Overweight	0.636172	0.596552	0.672414
	Obesity	0.487099	0.454733	0.519547

Table 16: Type II Error.

Feature	Level	Best Threshold	Accuracy
CAEC	Insufficient Weight	2	0.8479393652
CAEC	Normal Weight	2	0.8360966367
CAEC	Overweight	0	0.2747513027
CAEC	Obesity	1	0.4827096163
CALC	Insufficient Weight	0	0.1288488868
CALC	Normal Weight	2	0.8484130744
CALC	Overweight	2	0.7247749882
CALC	Obesity	1	0.5386072951
MTRANS	Insufficient Weight	3	0.3102794884
MTRANS	Normal Weight	4	0.8678351492
MTRANS	Overweight	0	0.2747513027
MTRANS	Obesity	3	0.4864992894
Gender	Insufficient Weight	0	0.1288488868
Gender	Normal Weight	1	0.4964471814
Gender	Overweight	1	0.5338702037
Gender	Obesity	0	0.4604452866
FAVC	Insufficient Weight	0	0.1288488868
FAVC	Normal Weight	0	0.1359545239
FAVC	Overweight	0	0.2747513027
FAVC	Obesity	1	0.5585030791
family_history_with_overweight	Insufficient Weight	0	0.1288488868
family_history_with_overweight	Normal Weight	0	0.1359545239
family_history_with_overweight	Overweight	1	0.3633349124
family_history_with_overweight	Obesity	1	0.6352439602
SCC	Insufficient Weight	1	0.8465182378
SCC	Normal Weight	1	0.8469919469
SCC	Overweight	1	0.7186167693
SCC	Obesity	0	0.4604452866
SMOKE	Insufficient Weight	0	0.1288488868
SMOKE	Normal Weight	1	0.8555187115
SMOKE	Overweight	0	0.2747513027
SMOKE	Obesity	1	0.5395547134

Table 17: Threshold and accuracy by feature and obesity level

Feature	Level	Sensitivity	Specificity	Kappa
CAEC	Insufficient Weight	0.4522	0.9065	0.3462
CAEC	Normal Weight	0.4111	0.9030	0.3105
CAEC	Overweight	1.0000	0.0000	0.0000
CAEC	Obesity	0.9979	0.0430	0.0379
CALC	Insufficient Weight	1.0000	0.0000	0.0000
CALC	Normal Weight	0.0662	0.9715	0.0552
CALC	Overweight	0.0603	0.9765	0.0506
CALC	Obesity	0.7562	0.3529	0.1051
MTRANS	Insufficient Weight	0.8309	0.2333	0.0205
MTRANS	Normal Weight	0.1115	0.9868	0.1488
MTRANS	Overweight	1.0000	0.0000	0.0000
MTRANS	Obesity	0.7840	0.2327	0.0158
Gender	Insufficient Weight	1.0000	0.0000	0.0000
Gender	Normal Weight	0.5087	0.4945	0.0015
Gender	Overweight	0.5724	0.5193	0.0727
Gender	Obesity	1.0000	0.0000	0.0000
FAVC	Insufficient Weight	1.0000	0.0000	0.0000
FAVC	Normal Weight	1.0000	0.0000	0.0000
FAVC	Overweight	1.0000	0.0000	0.0000
FAVC	Obesity	0.9805	0.1984	0.1676
family_history_with_overweight	Insufficient Weight	1.0000	0.0000	0.0000
family_history_with_overweight	Normal Weight	1.0000	0.0000	0.0000
family_history_with_overweight	Overweight	0.8293	0.1868	0.0100
family_history_with_overweight	Obesity	0.9918	0.3310	0.3054
SCC	Insufficient Weight	0.0809	0.9598	0.0561
SCC	Normal Weight	0.1045	0.9638	0.0950
SCC	Overweight	0.0707	0.9641	0.0469
SCC	Obesity	1.0000	0.0000	0.0000
SMOKE	Insufficient Weight	1.0000	0.0000	0.0000
SMOKE	Normal Weight	0.0453	0.9830	0.0440
SMOKE	Overweight	1.0000	0.0000	0.0000
SMOKE	Obesity	0.0226	0.9807	0.0036

Table 18: Sensitivity, specificity and Cohen's kappa by feature and obesity level

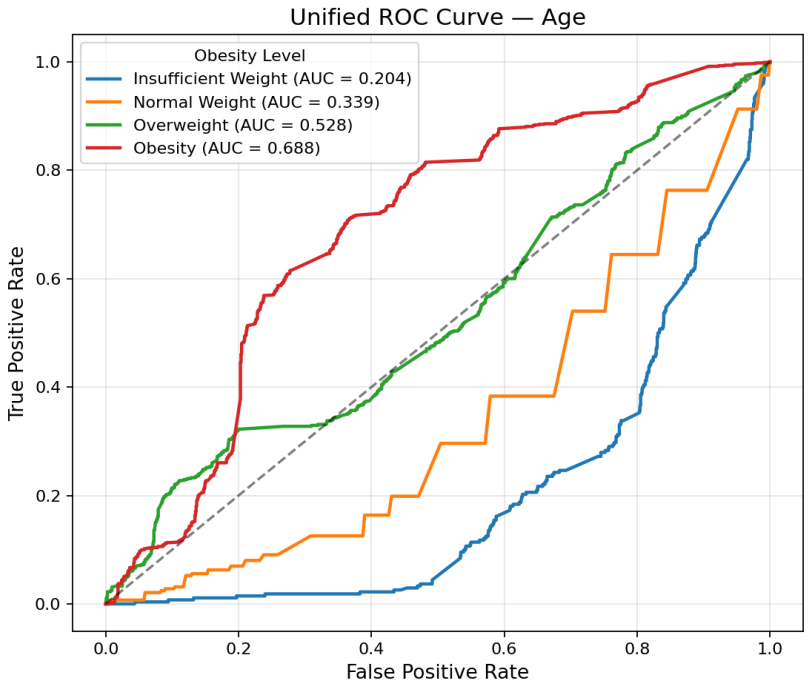

Figure 24: ROC curve for Age predicting Level 1 (one-vs-rest).

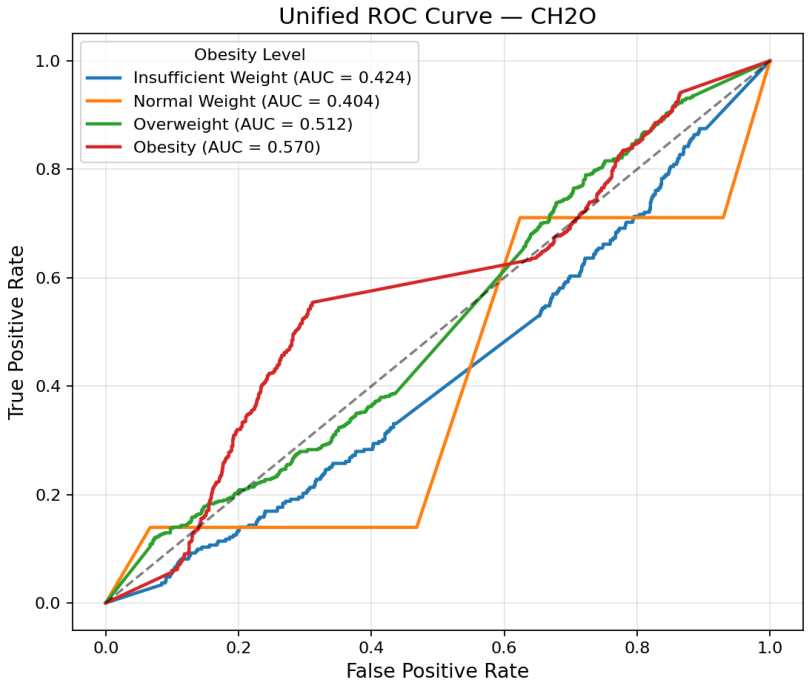

Figure 25: ROC curve for CH2O predicting Level 1 (one-vs-rest).

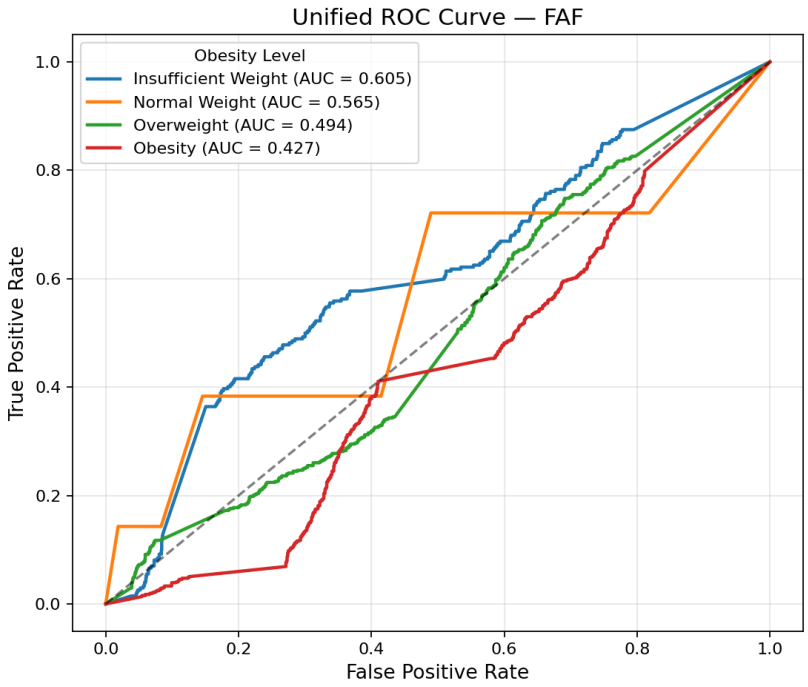

Figure 26: ROC curve for FAF predicting Level 1 (one-vs-rest).

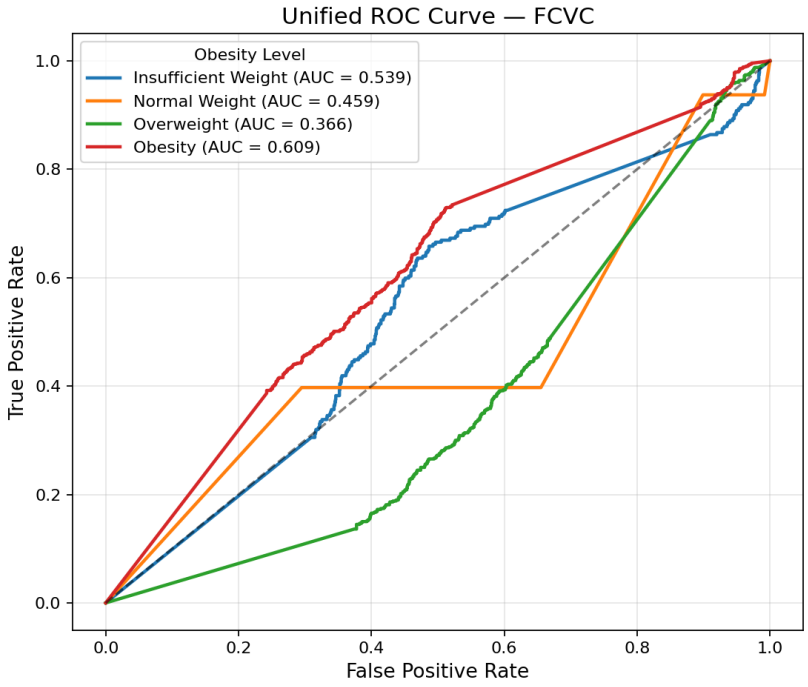

Figure 27: ROC curve for FCVC predicting Level 1 (one-vs-rest).

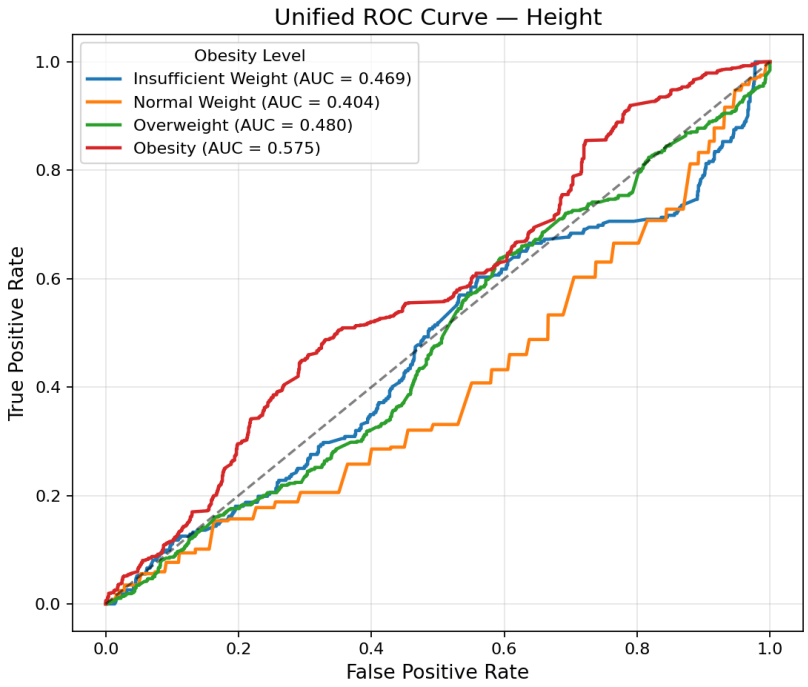

Figure 28: ROC curve for Height predicting Level 1 (one-vs-rest).

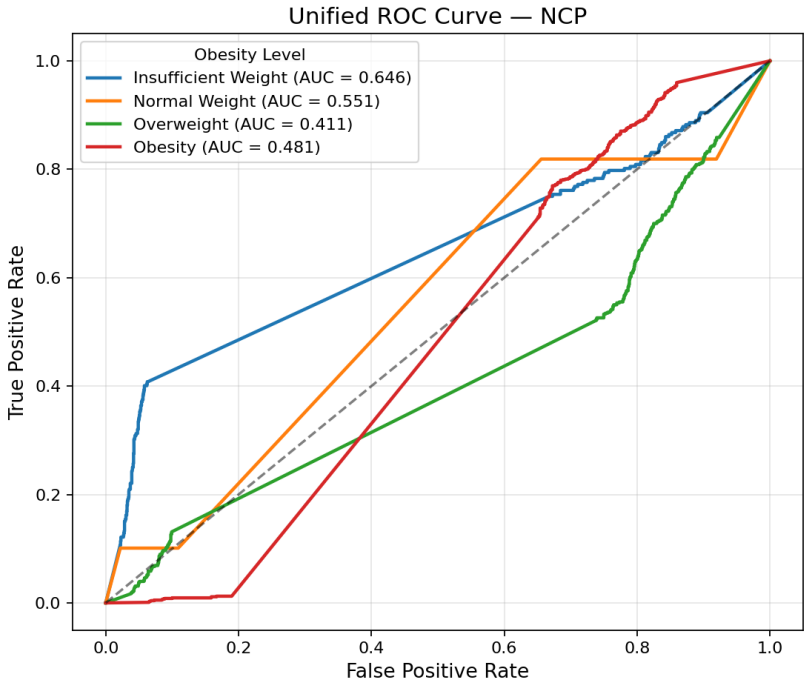

Figure 29: ROC curve for NCP predicting Level 1 (one-vs-rest).

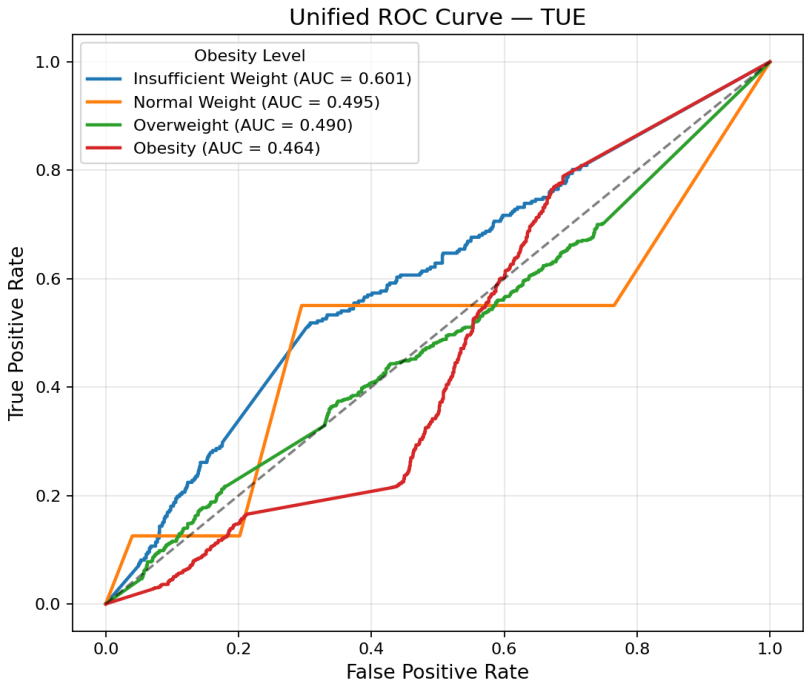

Figure 30: ROC curve for TUE predicting Level 1 (one-vs-rest).

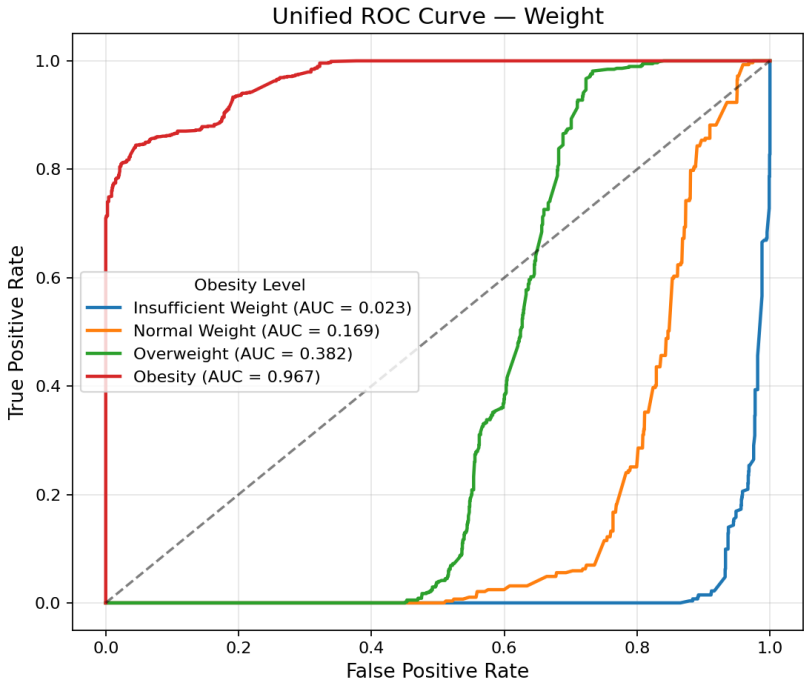

Figure 31: ROC curve for Weight predicting Level 1 (one-vs-rest).