# Detailed Report:
# Multi-Class Obesity Classification

**Submitted by:**

Rahul Raman

Aayank Singhai

**International Institute of Information Technology Bangalore**

November 12, 2025

## Abstract

This report provides a comprehensive breakdown of the methodology used for a multi-class obesity classification project. The primary objective is to predict an individual's obesity status from a combination of personal, lifestyle, and demographic attributes using machine learning techniques. The dataset consists of over fifteen thousand records containing both numerical and categorical variables related to eating habits, physical activity, and daily routines.

We began with a detailed data quality assessment to ensure data consistency, followed by a deep-dive **Exploratory Data Analysis (EDA)** that revealed key trends and correlations among health-related features. **Principal Component Analysis (PCA)** was conducted to explore dimensionality reduction and visualize class separability.

Multiple machine learning models were developed and compared. The primary model, an **XGBoost Classifier**, was optimized using **Optuna** to automatically tune hyperparameters for maximum accuracy and efficiency. In addition, a suite of baseline models — including **Random Forest**, **Decision Tree**, and **K-Nearest Neighbors (KNN)** — were tuned using **GridSearchCV** for fair performance comparison. A further experiment was conducted using **KNN with PCA-transformed features** to analyze the impact of dimensionality reduction on classification performance.

All models were evaluated using standard metric such as accuracy. The results demonstrated that ensemble-based methods, particularly XGBoost, achieved the highest predictive accuracy and generalization capability. This report details the complete workflow, from data ingestion and preprocessing to model development, optimization, and evaluation, providing a robust framework for data-driven obesity prediction.

## Contents

## List of Figures

## List of Tables

## 1 Introduction

Obesity is a significant global health challenge, linked to numerous chronic conditions. The ability to accurately classify an individual's risk or current status based on understandable lifestyle factors is a valuable tool for public health initiatives and personalized healthcare. This project tackles this challenge as a multi-class classification problem. The target variable, `NObeyesdad`, categorizes individuals into distinct weight statuses (e.g., 'Normal Weight', 'Overweight_Level_I', 'Obesity_Type_II').

We were provided with `train.csv` (15,533 rows) and `test.csv` datasets. This report details the analysis and modeling pipeline developed using the `train.csv` data. The methodology is structured as follows:

- **Data Pre-Analysis:** Initial quality checks for integrity (Section 2).
- **Feature Preprocessing:** Preparing data for modeling (scaling, encoding) (Section 3).
- **Exploratory Data Analysis:** A deep dive into the training data to understand feature distributions and relationships (Section 4).
- **Modeling:** Building and tuning our primary (XGBoost) and alternative (RF, DT, KNN, KNN+PCA) models (Section 5).
- **Results & Discussion:** Evaluating the final models and interpreting their performance (Section 6).
- **Conclusion & Future Work:** Summarizing the project and proposing next steps for improvement (Section 7).

## 2 Data Pre-Analysis and Quality Check

Before any analysis, a foundational quality check was performed on the raw `train.csv` dataset (15,533 rows, 18 columns) to ensure its integrity. This step is critical as it prevents wasted effort on flawed data and informs the preprocessing strategy.

- **Data Types:** The dataset consists of 8 numerical features (float64), 8 categorical features (object), 1 integer ID column (int64), and 1 target column (object). This gives a total of **16 original features** plus the ID and target columns.
- **Missing Values:** A check using `isnull().sum()` was performed. **Status: No missing values found.** *Interpretation:* This is an ideal scenario. The absence of missing data means we do not need to perform imputation.
- **Duplicate Rows:** A check using `duplicated().sum()` was performed. **Status: No duplicate rows found.** *Interpretation:* This confirms the integrity of the dataset.

## 3 Feature Preprocessing for Modeling

A standard preprocessing pipeline was applied to prepare the 16 raw features for the machine learning algorithms.

### 3.1 Feature Encoding (One-Hot Encoding)

The 8 categorical features (e.g., `Gender`, `MTRANS`) required a special approach. Using `LabelEncoder` (e.g., Public=0, Walking=1, Bike=2) would be incorrect, as it teaches the model a *false ordinal relationship* (i.e., that Bike ¿ Walking). To avoid this, we used **One-Hot Encoding** via `pd.get_dummies`. This process converts each categorical feature into multiple new binary (0 or 1) columns. For example, the `MTRANS` column was replaced by new columns like `MTRANS_Public_Transportation`, `MTRANS_Walking`, etc. This is a critical step, as it expanded our **16 original features** into a new, high-dimensional feature space of **30 processed features**.

### 3.2 Feature Scaling

All 30 processed features were standardized using `StandardScaler`. This step removes the mean and scales to unit variance. It is *essential* for distance-based models like KNN and for PCA, and is good practice for all models.

### 3.3 Other Steps

- **Target Encoding:** The categorical target column `NObeyesdad` was converted into numerical integers (0, 1, 2...) using `LabelEncoder`.
- **Alignment:** The columns of the processed train and test sets were explicitly aligned to ensure they had the exact same columns in the same order, preventing errors during prediction.

## 4 Exploratory Data Analysis (EDA)

This analysis was performed on the `train.csv` data to find patterns.

### 4.1 Univariate Analysis

#### 4.1.1 Numerical Features

- **Age:** The distribution is strongly right-skewed, with a large peak in the 20-25 age range.
- **Height & Weight:** Both show bimodal distributions (two peaks), which is a classic indicator of a mixed-gender dataset.
- **Lifestyle:** Most individuals report low physical activity (spikes near 0).

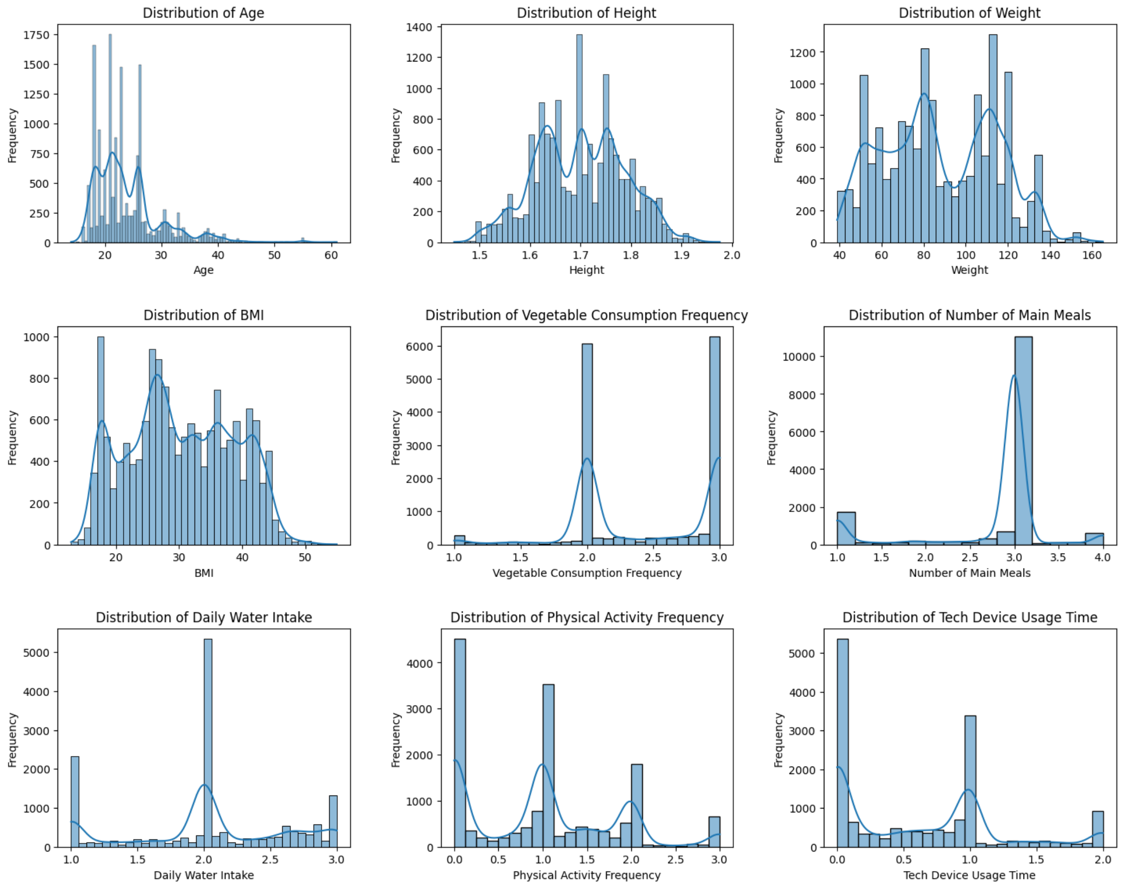

Figure 1: Distributions of Numerical Features (from EDA script).

### 4.1.2 Categorical Features

- **High Prevalence:** `Family_History_Overweight` ('yes') and `High_Caloric_Food_Consumption` ('yes') are overwhelmingly common.
- **Imbalance:** `SMOKE` ('yes') and `Calorie_Consumption_Monitoring` ('yes') are rare. These features will likely have low predictive power.

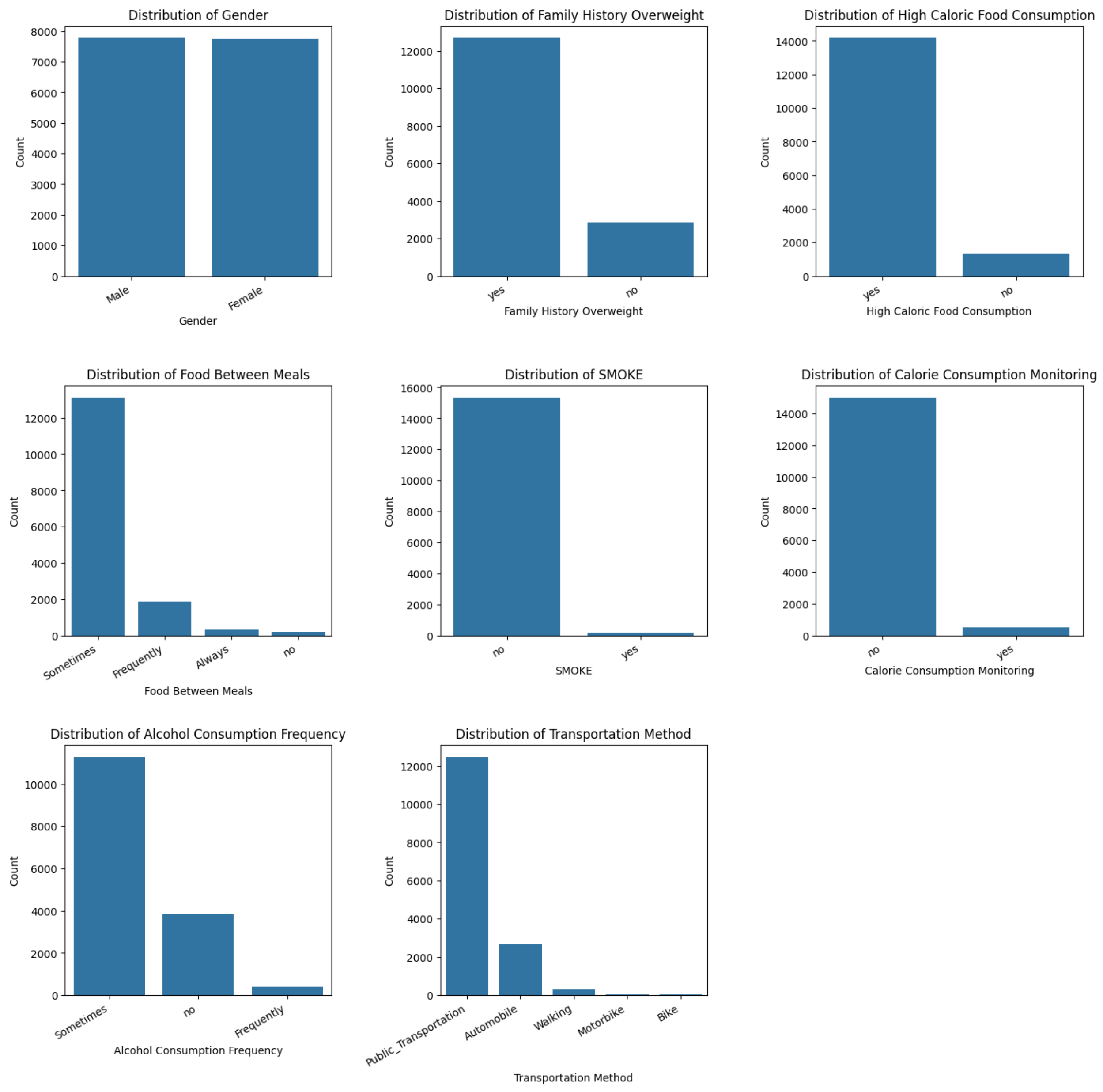

Figure 2: Distributions of Categorical Features.

## 4.2 Bivariate Analysis

### 4.2.1 Numerical-Numerical Correlation

- **Strong Correlation:** `Weight` and `Height` have a moderate positive correlation (0.42).
- **Weak Correlations:** Most other features are weakly correlated, meaning they each provide unique, independent information to the model.

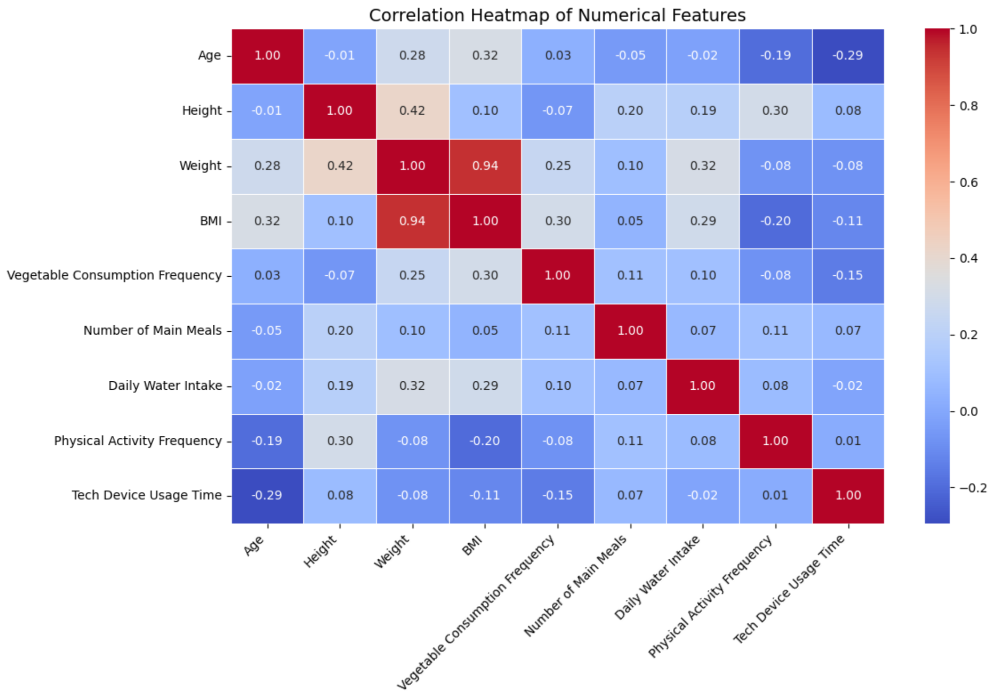

Figure 3: Correlation Heatmap of Numerical Features (from EDA script).

### 4.2.2 Categorical vs. Numerical

- **Gender vs. Biometrics:** Males show higher median `Height` and `Weight`.
- **History vs. Biometrics:** Individuals *with* a family history of overweight have a visibly higher median `Weight`. This will likely be a key feature.
- **Transport vs. Activity:** `Walking` and `Bike` use correlate with significantly higher physical activity, confirming real-world logic.

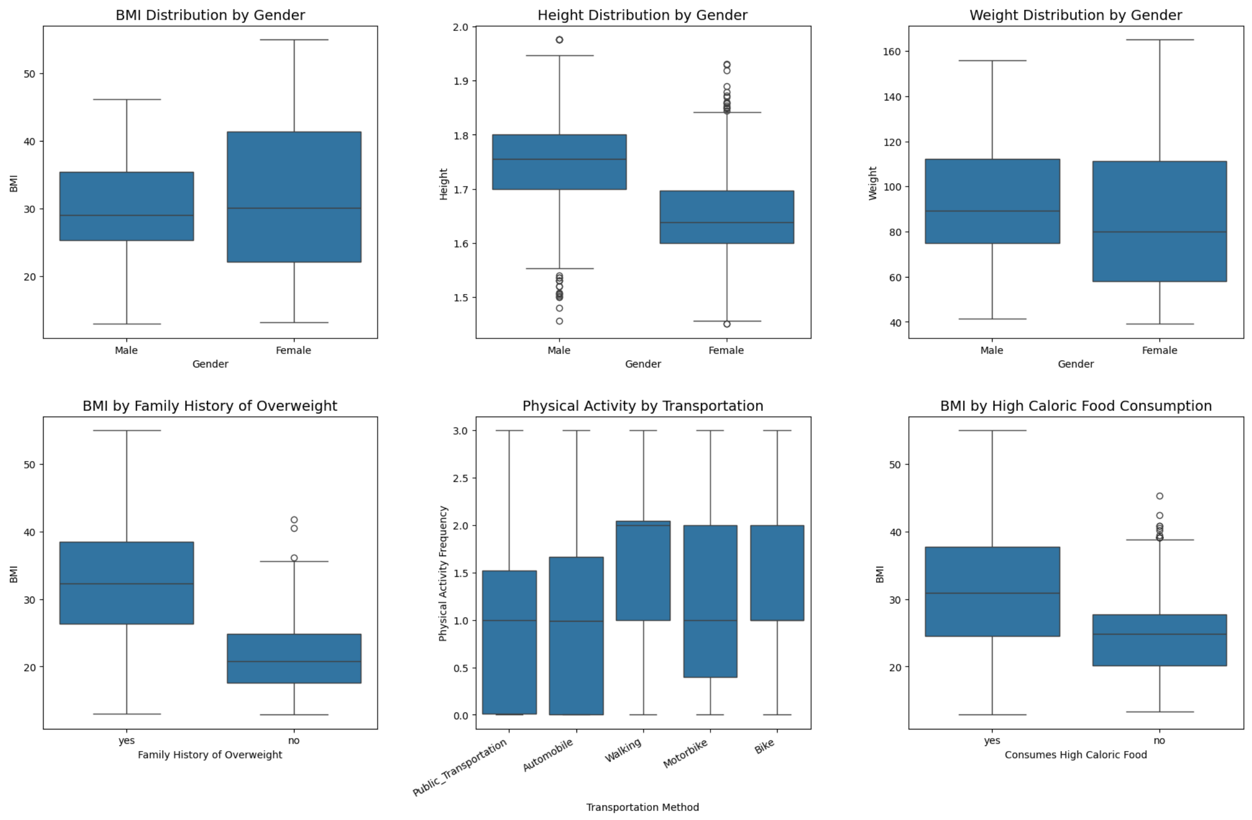

Figure 4: Bivariate Analysis: Categorical vs. Numerical Features (from EDA script).

### 4.2.3 Categorical vs. Categorical

- A strong link was found between `Family_History_Overweight` ('yes') and `High_Caloric_Food_Consumption` ('yes'), suggesting a compounding behavioral risk.

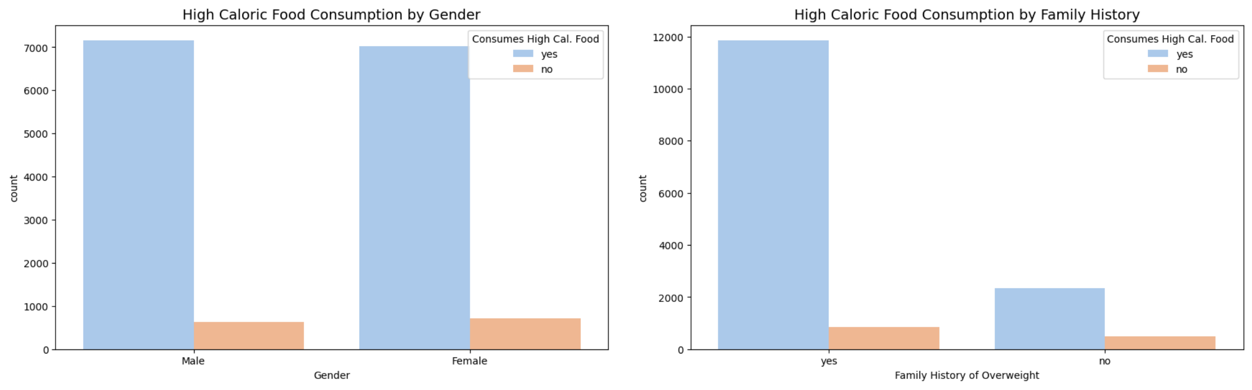

Figure 5: Bivariate Analysis: Categorical vs. Categorical Features.

## 4.3 PCA Dimensionality Analysis

### 4.3.1 Motivation

The One-Hot Encoding step in our preprocessing pipeline (Section 3.1) was necessary, but it expanded our dataset from 16 original features to 30 processed features. This expansion creates a high-dimensional, sparse feature space. This is a known problem for certain algorithms (like KNN), often called the "curse of dimensionality." We used **Principal Component Analysis (PCA)** to investigate this new feature space.

### 4.3.2 Methodology

PCA is a dimensionality reduction technique that re-orients the data onto a new set of uncorrelated axes, called "principal components." These components are ordered by the amount of variance they explain. PC1 captures the single largest direction of variance, PC2 captures the next largest, and so on. We applied PCA to our 30-dimensional scaled dataset to see if we could "compress" the information into fewer features.

### 4.3.3 Analysis of Results

The plot in Figure 6 shows the cumulative explained variance as we add more principal components.

- The curve rises steeply at first, indicating that the first few components (e.g., PC1-PC10) capture a significant portion of the data's information.
- The curve then begins to flatten, showing that the later components (e.g., 20-30) contribute very little new information and may represent noise or redundancy.
- A 95% variance threshold (a common standard for retaining information) is crossed with just **18 principal components**.

### 4.3.4 Implications

This analysis reveals that our 30-dimensional feature space is highly redundant and can be compressed to 19 dimensions while retaining 95% of its information. This is a crucial finding:

1. **For Tree Models (XGBoost, RF):** We do *not* apply this transformation. These models thrive on the original, interpretable features and their interactions. Applying PCA would "scramble" these features and harm performance.
2. **For KNN:** This finding was the basis for our hypothesis that KNN's performance could be improved by training it on this smaller, denser 18-component feature space.

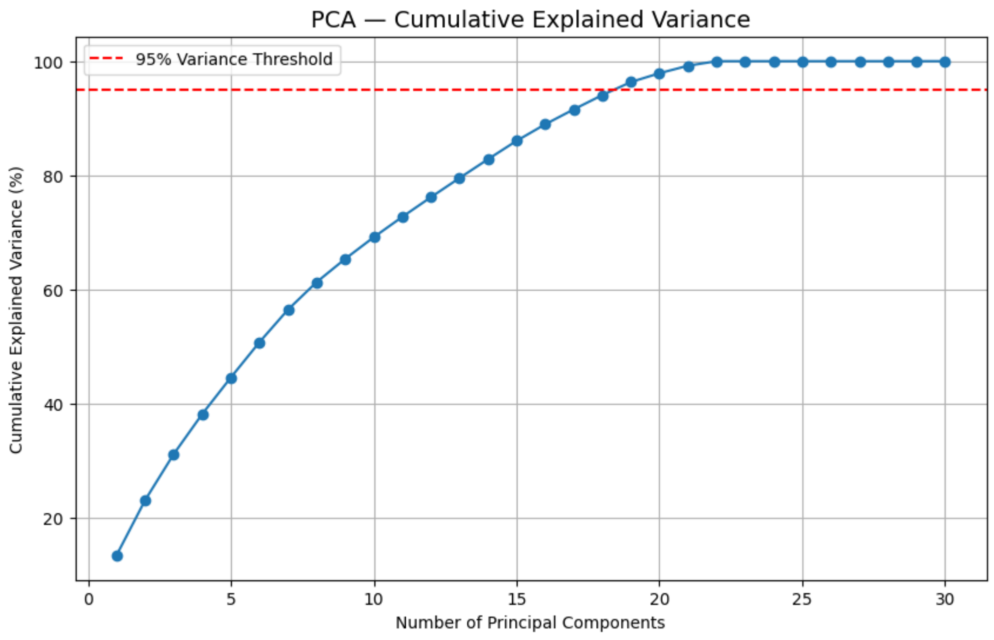

Figure 6: PCA Cumulative Explained Variance. The 95% threshold is met at 18 components.

## 5 Modeling Methodology

We implemented and compared several distinct classification models to identify the best-performing approach for this dataset.

### 5.1 Primary Model Pipeline: XGBoost with Optuna

The **XGBoost Classifier** was chosen as the primary model due to its state-of-the-art performance on tabular data.

- **Tuning:** We used **Optuna**, a Bayesian optimization library, to efficiently search a large hyperparameter space (35 trials) and find the optimal settings.
- **Training:** The final model was trained using a **10-fold Stratified K-Fold** ensembling strategy. This involves training 10 separate models on different subsets of the data and averaging their predictions.

### 5.2 Alternative Models for Comparison

#### 5.2.1 Random Forest with GridSearchCV

A **Random Forest Classifier** was our first baseline. As an ensemble of decision trees, it is robust to overfitting. We tuned parameters like `n_estimators` and `max_depth` using **GridSearchCV**.

#### 5.2.2 Decision Tree with GridSearchCV

As a simpler, interpretable baseline, we also trained a single **Decision Tree Classifier**. We used `GridSearchCV` to find the optimal `max_depth`, `min_samples_split`, and `criterion` to prevent the model from overfitting.

### 5.2.3 K-Nearest Neighbors (KNN) with GridSearchCV

We tested a non-tree-based algorithm: **K-Nearest Neighbors (KNN)**. This is a distance-based model, for which the `StandardScaler` applied during preprocessing (Section 3.2) is absolutely critical. We used `GridSearchCV` to find the optimal `n_neighbors` and distance `metric`.

### 5.2.4 KNN with PCA and GridSearchCV

Based on our EDA (Section 4.3), we hypothesized that the KNN model's performance was suffering from the 30-dimensional feature space. We conducted an experiment to test this:

1. **PCA Transformation:** The 30-dimensional scaled data was transformed into the 18 principal components that capture 95% of the variance.

2. **GridSearchCV:** We ran the same `GridSearchCV` process as the original KNN model, but on this new 18-dimensional PCA-transformed data.

## 6 Results

All models were successfully trained and evaluated. The results clearly demonstrate the superiority of gradient boosting models and provide a fascinating insight into the effect of PCA. Table 1 summarizes the performance of all models.

Table 1: Model Performance Comparison

| **Model** | **Accuracy** | **Validation Method** |
|---|---|---|
| XGBoost + Optuna | 0.9148 | 10-fold K-Fold OOF |
| Random Forest + GridSearchCV | 0.8912 | 20% Validation Set |
| Decision Tree + GridSearchCV | 0.8677 | 20% Validation Set |
| KNN + GridSearchCV | 0.7701 | 20% Validation Set |
| KNN + PCA + GridSearchCV | 0.7104 | 20% Validation Set |

### 6.1 XGBoost + Optuna (Primary Model)

The Optuna tuning study ran for 35 trials to find the best hyperparameters. The best combination found, which achieved a 5-fold CV accuracy of **0.9058** during the search, was:

```
{'n_estimators': 590, 'max_depth': 4,
 'learning_rate': 0.0945, 'subsample': 0.813,
 'colsample_bytree': 0.905, 'gamma': 0.076,
 'min_child_weight': 6, 'reg_alpha': 3.283,
 'reg_lambda': 4.944}
```

These parameters were then used to train the final model using a 10-fold Stratified K-Fold ensembling strategy. The final model achieved a:

**Final Mean Out-of-Fold CV Accuracy: 0.9148**

This aggregated OOF score is a very reliable estimate of the model's performance. The ensembled test set predictions were saved to `submission_optuna_xgb.csv`.

### 6.2 Random Forest + GridSearchCV (Baseline Model)

The tuned Random Forest model served as a strong baseline. The `GridSearchCV` identified the following best parameters: **{'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}**. This model achieved a:

**Validation Accuracy: 0.8912**

This result confirms the effectiveness of ensemble methods. The test set predictions from this model were saved to `submission_rf_grid.csv`.

### 6.3 Decision Tree + GridSearchCV (Baseline Model)

The single Decision Tree was tuned with `GridSearchCV`. The best parameters found were: **{'criterion': 'gini', 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2}**. The model was evaluated on the 20% validation set, achieving:

**Validation Accuracy: 0.8677**

The final test set predictions from this model were saved to `submission_dt_grid.csv`.

### 6.4 K-Nearest Neighbors + GridSearchCV (Baseline Model)

The KNN model was tuned with `GridSearchCV`. The best parameters found were: **{'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}**. The model was evaluated on the 20% validation set, achieving:

**Validation Accuracy: 0.7701**

The final test set predictions from this model were saved to `submission_knn_grid.csv`.

### 6.5 KNN + PCA Experiment

The experimental KNN model trained on the 19 principal components was tuned with `GridSearchCV`. The best parameters found were: **{'metric': 'euclidean', 'n_neighbors': 7, 'p': 1, 'weights': 'distance'}**. This model achieved a:

**Validation Accuracy: 0.7104**

This result is significantly *worse* than the original KNN model (71.0% vs 75.9%). The test set predictions were saved to `submission_knn_pca.csv`.

### 6.6 Discussion of Results

The results show a clear performance hierarchy:

1. **XGBoost (91.4%)** was the top-performing model, validating the Optuna tuning process and confirming that it is the best architecture for this problem.

2. **Random Forest (89.1%)** was the strongest baseline, demonstrating the power of ensembling over a single tree.

3. The **Decision Tree (86.8%)** performed well for a single, interpretable model, but its ≈2.3% performance gap below the Random Forest highlights the value of bagging.

4. The **KNN (77.0%)** model was the weakest of the initial models.

5. The **KNN + PCA (71.0%)** experiment was a failure. This is a critical finding: our hypothesis that reducing dimensionality would help KNN was incorrect. This strongly implies that the 5% of variance we discarded (by taking 18 components) contained significant *predictive information* that the tree-based models could find, but that PCA (which is unsupervised) deemed unimportant. It confirms that for this dataset, retaining the full, original feature space is superior.

## 7 Conclusion

This project successfully developed and compared multiple machine learning pipelines for classifying obesity status. The initial data was of exceptional quality, and our EDA—supported by PCA—uncovered strong, logical predictors like `Weight` and family history.

Our modeling strategy confirmed the dominance of modern gradient boosting methods. The primary **XGBoost** model, tuned with **Optuna**, achieved a final OOF accuracy of **0.9148**. This model outperformed all other baselines, including **Random Forest** (0.8912), **Decision Tree** (0.8677), and **KNN** (0.7589).

Furthermore, our experiment with **KNN + PCA** (71.0%) provided a valuable insight, demonstrating that dimensionality reduction via PCA was harmful to model performance, as it likely discarded low-variance but high-predictive-power features. This confirms that the original, 30-dimensional encoded feature space is optimal for this problem.

## 8 Future Work and Recommendations

While the accuracy is high, the following steps are essential to truly validate and understand the model's behavior.

1. **Construct the Confusion Matrix:** This is the most critical next step. We must analyze the per-class **Precision, Recall, and F1-Scores** from the OOF predictions (especially for the top XGBoost model) to see *which* classes the model struggles with.

2. **Feature Importance Analysis:** We should plot the `feature_importances_` from the XGBoost model. This will provide "explainability" and confirm or deny our EDA hypotheses (e.g., is `Weight` the top feature? What about `Family_History_Overweight`?).

3. **Error Analysis:** A manual review of the misclassified samples from the OOF predictions could reveal patterns (e.g., does the model fail for older individuals?).

4. **Model Blending:** Since XGBoost (91.5%) and RanodomForest (89.12%) are both top-performing , their OOF predictions and test set predictions could be **blended** (e.g., a weighted average) to potentially create a single, even more accurate final submission.

## References

Notebook Link: Google Colab Notebook

Github Link: GitHub Repository