Machine-Learning Project Report

Obesity Classification Using Optimized XGBoost Models

AIT511 Machine Learning

Group Name: MT2025058_MT2025710

Submitted By:

Kakadiya Aman Jitendrabhai
MT2025058

Koringa Yash Mansukhbhai
MT2025710

Under The Guidance Of:

Prof. Aswin Kannan
Department Of Computer Science And Engineering
International Institute Of Information Technology Bangalore

Project Repository:

https://github.com/AmanKakadiya301/ML-Project

Click here

Contents

Chapter 1

Introduction

The goal of this project is to build an optimized machine learning model to classify obesity levels based on demographic, physical, and lifestyle information. The dataset, sourced from Kaggle, contains attributes such as age, gender, height, weight, diet, physical activity, water intake, and technology usage.

To achieve reliable predictions, the **XGBoost** algorithm was employed and fine-tuned using **Optuna** for hyperparameter optimization. The workflow followed in this project is outlined below:

- **Data Preparation:** Imported and cleaned the dataset, ensuring consistent formatting and handling of missing values.
- **Feature Processing:** Encoded categorical features using XGBoost's native categorical support for better compatibility and performance.
- **Model Development:** Implemented XGBoost with GPU acceleration to enhance computational efficiency.
- **Hyperparameter Optimization:** Applied Optuna's Bayesian optimization to tune key parameters such as learning rate, tree depth, and regularization terms.
- **Model Evaluation:** Used 5-fold Stratified Cross-Validation to ensure model robustness and prevent overfitting.

After optimization, the XGBoost model demonstrated reliable performance in predicting obesity levels using demographic, lifestyle, and physiological characteristics.

Chapter 2

Data Processing

This chapter describes the procedures applied to prepare the dataset for model training and optimization. Effective data processing ensures the model operates efficiently and produces reliable predictions.

2.1 Dataset Overview

The training dataset (`train.csv`) contains **15,533 records** and **18 features**, including demographic, physical, and lifestyle attributes, along with the target variable (`WeightCategory`). A preliminary inspection confirmed that:

- The dataset has **no missing values**, ensuring data completeness.
- No imputation or record removal was required.
- The data is balanced and well-structured for classification tasks.

2.2 Loading the Data

The dataset was imported from CSV files for both training and testing purposes. During the loading process:

- The `id` column was removed as it does not contribute to prediction.
- The target column (`WeightCategory`) was identified automatically and separated from the input features.
- The features were organized into two sets: `X_full` for training and `X_test` for evaluation.

2.3 Feature Preparation and Encoding

XGBoost (version 2.x) supports **native categorical encoding**, eliminating the need for traditional One-Hot Encoding. This feature was used to simplify preprocessing and maintain model interpretability.

- All **categorical features** were converted to the `category` data type to enable XGBoost's internal categorical handling.
- Categories were aligned between the training and test sets to avoid mismatched labels.
- The **target variable** was encoded numerically using **Label Encoding**, where each obesity class was assigned a unique integer value for multi-class classification.

2.4 Device Configuration

To maximize computational efficiency, the model automatically selected the appropriate hardware:

- If a **CUDA-enabled GPU** was available, training was performed on the GPU for faster processing.
- Otherwise, the model defaulted to the **CPU** using optimized histogram-based tree methods.

2.5 Summary

After preprocessing:

- The dataset was confirmed to be clean, consistent, and complete.
- All categorical variables were encoded using XGBoost's native categorical support.
- The target variable was transformed into numeric labels using Label Encoding.
- The environment was configured for either GPU or CPU execution, ensuring efficient model training.

These steps ensured the dataset was fully prepared for subsequent model training, hyperparameter optimization, and evaluation using the XGBoost and Optuna frameworks.

Chapter 3

EDA (Exploratory Data Analysis)

Exploratory Data Analysis (EDA) is an important process used to understand the underlying patterns, distributions, and relationships present in a dataset. It helps identify trends, detect anomalies, and uncover dependencies that can guide the model-building phase and improve predictive performance.

3.1 Distribution of the Target Variable

The distribution of the target variable *WeightCategory* is shown in Figure 3.1. The plot indicates that the dataset is imbalanced: the *Normal Weight* and *Overweight* categories constitute the majority of samples, while *Underweight* and *Obesity Type III* are relatively rare. This imbalance should be considered when training classification models.

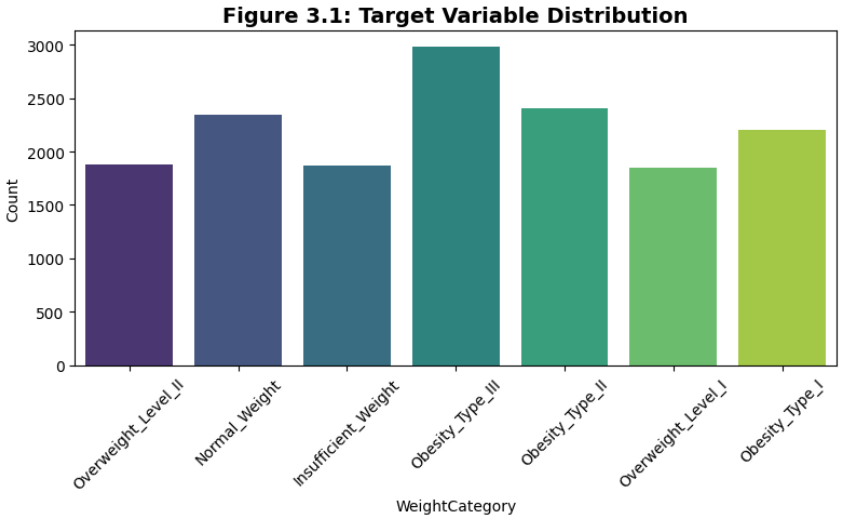

Figure 3.1: Distribution of Target Variable

3.2 Categorical Feature Distribution

Categorical variables such as *Gender*, *Family History*, and *Smoking Habits* show distinctive patterns across obesity classes (Figure 3.2). Participants with a family history of obesity are more likely to fall into higher obesity categories, whereas smoking habits show minimal variation across classes.

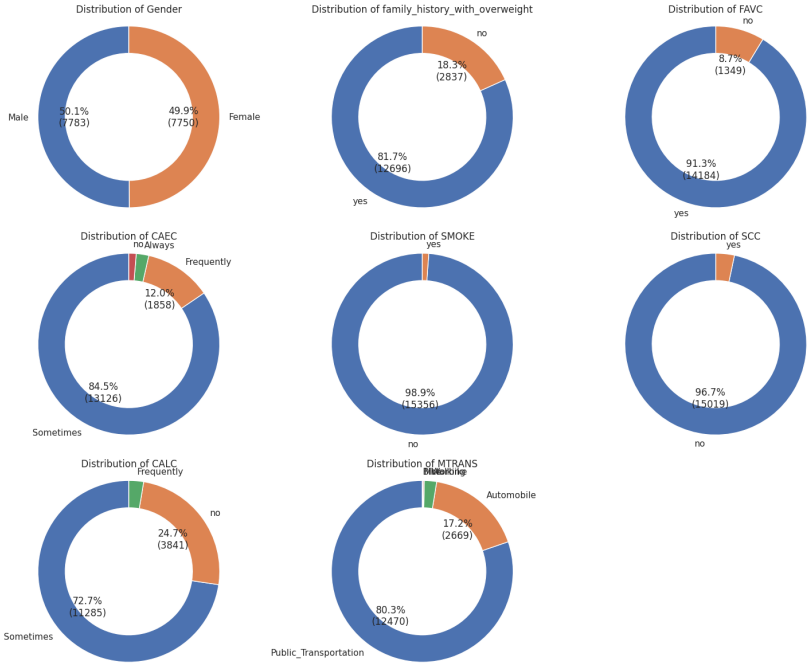

Figure 3.2: Distribution of Categorical Features

3.3 Numerical Feature Distribution

Continuous variables such as *Age*, *Height*, and *Weight* exhibit distinct statistical patterns. *Weight* follows a right-skewed distribution, while *Height* appears approximately normal. Most participants are between the ages of 20 and 35, suggesting that the dataset primarily represents a young adult population.

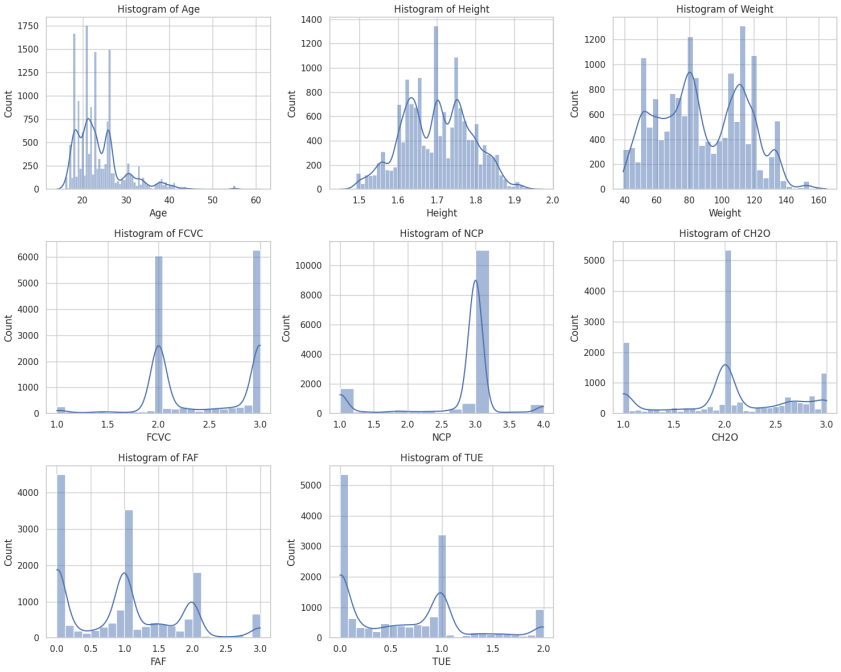

Figure 3.3: Distribution of Numerical Features

3.4 Weight vs Obesity Level

Figure 3.4 compares *Weight* across obesity categories using boxplots. It is evident that average weight increases progressively from underweight to obese categories. Additionally, obese individuals tend to be slightly older (around 30 years), while participants below 20 years are more likely to belong to the underweight or normal-weight groups.

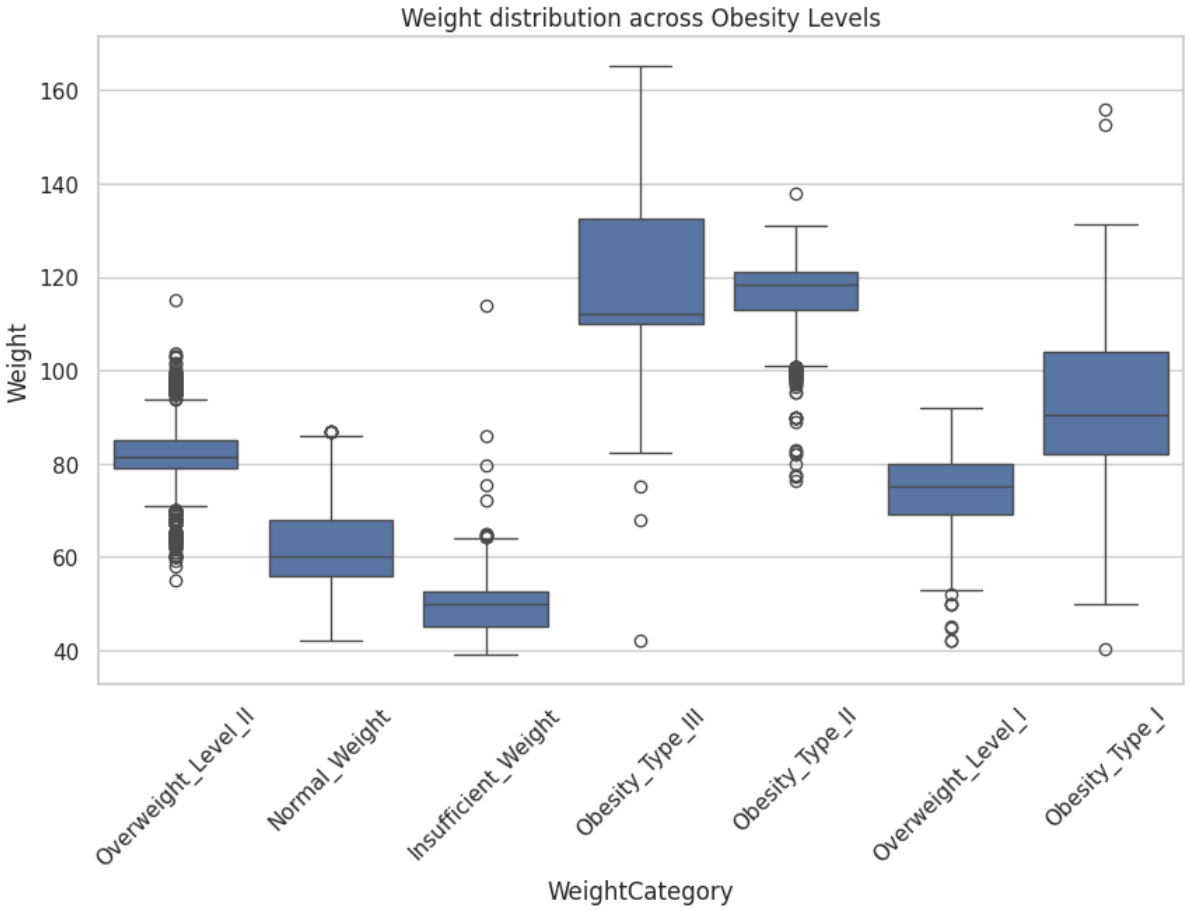

Figure 3.4: Weight vs Obesity Level

3.5 Correlation Heatmap

The correlation heatmap (Figure 3.5) shows a strong positive correlation between *Weight* and obesity level. Conversely, *FAF* (Frequency of Physical Activity) is negatively correlated, indicating that higher physical activity levels are associated with lower obesity risk. Other features show moderate relationships, contributing collectively to obesity prediction.

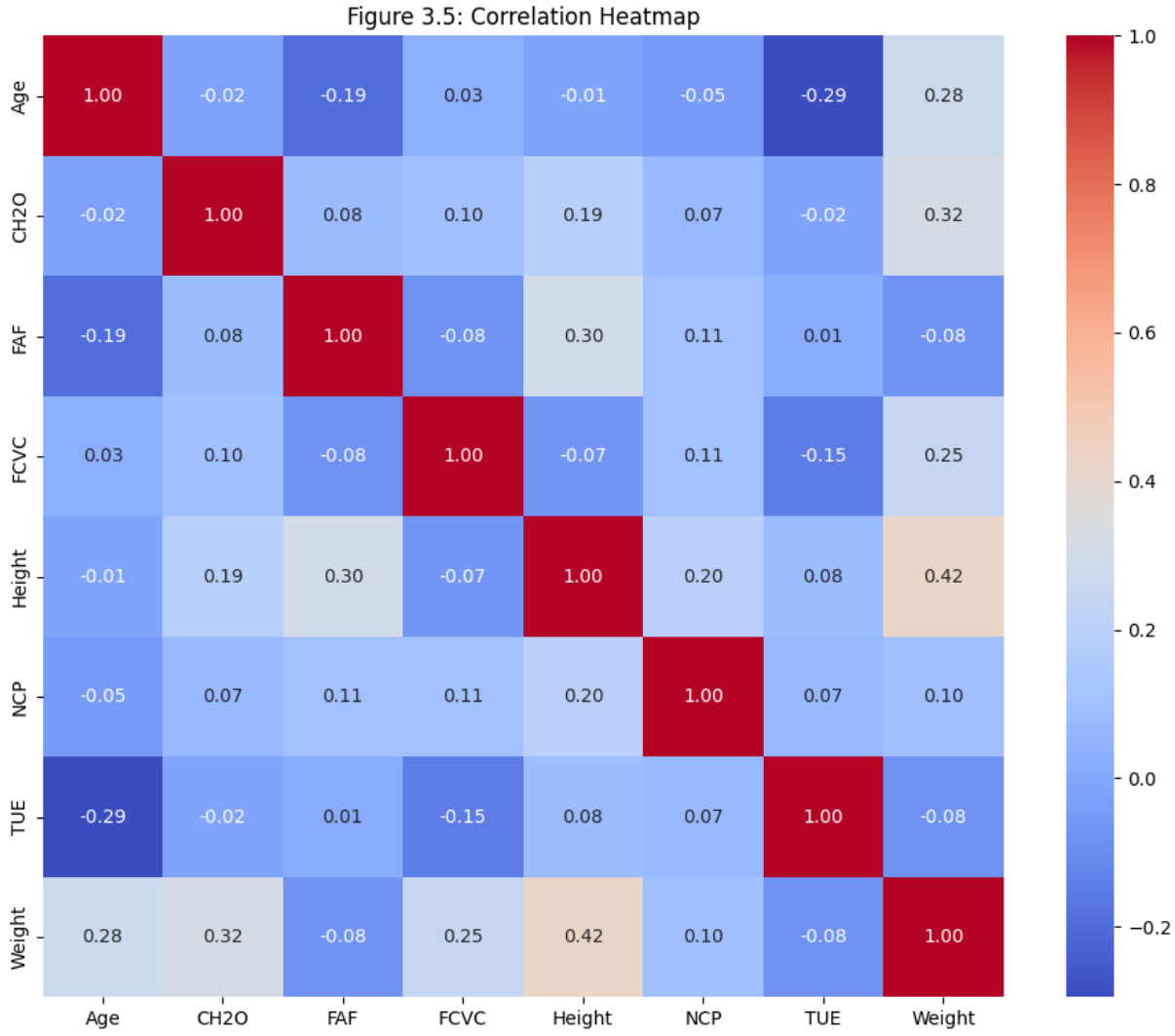

Figure 3.5: Correlation Heatmap of Numerical Features

3.6 Box Plots and Pairplots

Boxplots of *Weight* versus *WeightCategory* demonstrate a consistent increase in median weight with higher obesity levels. Pairplots of *Height*, *Weight*, and *Age* reveal visually separable clusters corresponding to different obesity classes. These observations indicate that the dataset possesses strong predictive features suitable for classification using machine learning algorithms such as XGBoost.

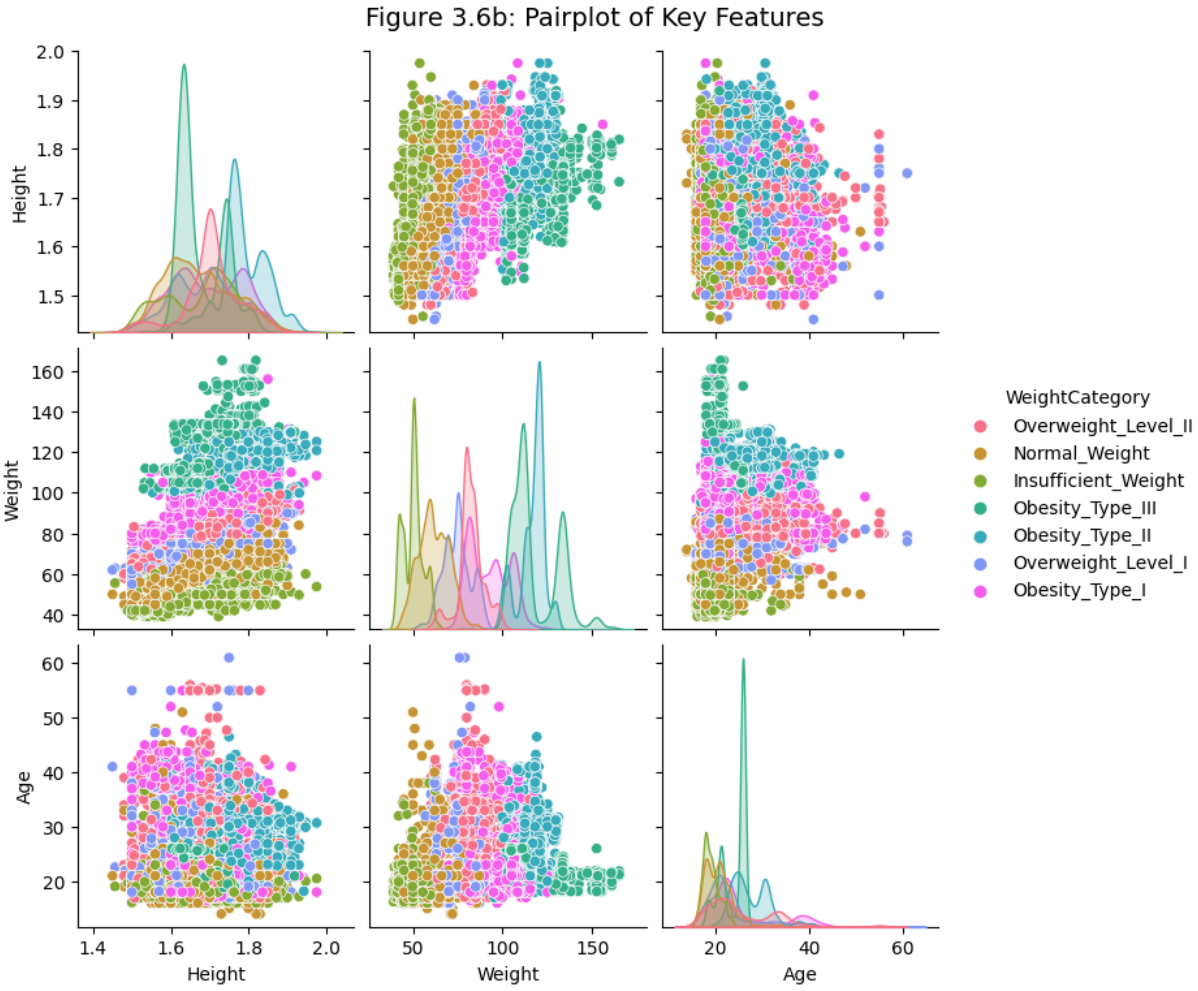

Figure 3.6: Box Plots and Pairplots of Key Features

Chapter 4

Implementation of Machine Learning Models

This chapter discusses the implementation and evaluation of nine machine learning algorithms applied to obesity level classification. Each model was trained on preprocessed data and assessed using metrics such as accuracy, precision, recall, and F1-score. Among all models, XGBoost achieved the highest cross-validation accuracy of 90.31%, followed closely by Gradient Boosting and Random Forest. The following sections describe each model's functionality, advantages, limitations, and experimental performance.

4.1 Logistic Regression

Logistic Regression is a statistical model that predicts categorical outcomes using a logistic function. It serves as a strong baseline model due to its simplicity and interpretability.

Advantages:

- Simple and efficient for linearly separable data.
- Interpretable coefficients that provide insight into feature influence.
- Fast to train and performs well on small to medium-sized datasets.

Limitations:

- Limited capability for capturing non-linear relationships.
- Sensitive to outliers and multicollinearity.

Performance: Achieved a validation accuracy of 85.58% with a cross-validation score of 85.20% ($\pm$ 0.34%).

4.2 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm that classifies samples based on the majority label among their nearest neighbors.

Advantages:

- Simple and intuitive algorithm.
- No training phase; suitable for small datasets.

- Naturally handles multi-class problems.

Limitations:

- Computationally expensive for large datasets.
- Sensitive to irrelevant features and data scaling.

Performance: Achieved a validation accuracy of 72.87% and a cross-validation score of 73.19% (± 0.82%).

4.3 Decision Tree

The Decision Tree model divides the dataset based on feature thresholds, creating a hierarchy of decisions for classification.

Advantages:

- Easy to interpret and visualize.
- Captures non-linear relationships between features.

Limitations:

- Prone to overfitting without pruning or ensemble methods.
- Sensitive to small data variations.

Performance: Achieved a validation accuracy of 86.19% and a cross-validation score of 86.38% (± 0.81%).

4.4 Naive Bayes

Naive Bayes applies Bayes' theorem with the assumption of feature independence. It is particularly effective for categorical data.

Advantages:

- Extremely fast and efficient.
- Performs well even with limited training data.

Limitations:

- Assumes independence between predictors, which may not hold true in practice.
- Lower accuracy for complex or correlated data.

Performance: Achieved a validation accuracy of 66.01% and a cross-validation score of 66.17% (± 0.94%).

4.5 Support Vector Machine (SVM)

SVM constructs hyperplanes that best separate data points of different classes. It is particularly effective for high-dimensional spaces.

Advantages:

- Effective for both linear and non-linear data (using kernel functions).
- Works well in high-dimensional feature spaces.

Limitations:

- Computationally expensive for large datasets.
- Requires careful parameter tuning and kernel selection.

Performance: Achieved a validation accuracy of 85.03% with a cross-validation score of 84.81% ($\pm$ 0.53%).

4.6 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees and combines their outputs to improve generalization.

Advantages:

- Naturally resistant to overfitting due to averaging across multiple trees.
- Handles both numerical and categorical data effectively.
- Provides feature importance rankings.

Limitations:

- Computationally intensive for large ensembles.
- Less interpretable compared to single decision trees.

Performance: Achieved a validation accuracy of 89.67% with a cross-validation score of 89.59% ($\pm$ 0.45%).

4.7 Gradient Boosting

Gradient Boosting constructs an ensemble of weak learners (decision trees) sequentially, where each tree corrects the errors of the previous one.

Advantages:

- Capable of modeling complex relationships.
- Provides strong predictive performance on structured data.

Limitations:

- Computationally expensive and prone to overfitting if not tuned.
- Sensitive to noisy data.

Performance: Achieved a validation accuracy of 90.15% and a cross-validation score of 89.95% ($\pm$ 0.54%).

4.8 XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting framework that uses advanced regularization and parallelization techniques.

Advantages:

- Handles missing values and outliers efficiently.
- Supports hyperparameter tuning and regularization for improved generalization.
- Offers high predictive accuracy and computational efficiency.

Limitations:

- Requires extensive hyperparameter tuning for optimal performance.
- Less interpretable compared to simpler models.

Performance: Achieved the highest validation accuracy of 90.31% with a cross-validation score of 90.28% (± 0.39%).

4.9 AdaBoost

AdaBoost (Adaptive Boosting) improves weak learners by focusing on previously misclassified instances, combining them to form a strong classifier.

Advantages:

- Enhances weak learners effectively.
- Relatively simple ensemble method.

Limitations:

- Sensitive to noisy data and outliers.
- May underperform on complex, non-linear data compared to XGBoost.

Performance: Achieved a validation accuracy of 65.98% and a cross-validation score of 64.67% (± 4.48%).

4.10 Model Comparison

Table 4.1 summarizes the performance metrics of all trained models. It is evident that XGBoost, Gradient Boosting, and Random Forest consistently outperform other algorithms in terms of validation accuracy and overall stability. [Note : Run time=Time(s)]

Table 4.1: Comparison of Model Performance Metrics

Model	Train Acc.	Val Acc.	CV Mean	CV Std	Precision	Recall	F1	Time (s)
XGBoost	0.9499	0.9031	0.9028	0.0039	0.9025	0.9031	0.9026	9.03
Gradient Boosting	0.9249	0.9015	0.8995	0.0054	0.9010	0.9015	0.9011	75.82
Random Forest	1.0000	0.8967	0.8959	0.0045	0.8959	0.8967	0.8960	10.53
Decision Tree	0.9195	0.8619	0.8638	0.0081	0.8603	0.8619	0.8607	0.39
Logistic Regression	0.8552	0.8558	0.8520	0.0034	0.8539	0.8558	0.8541	4.71
SVM	0.8939	0.8503	0.8481	0.0053	0.8496	0.8503	0.8499	21.80
KNN	0.8156	0.7287	0.7319	0.0082	0.7236	0.7287	0.7242	2.99
Naive Bayes	0.6638	0.6601	0.6617	0.0094	0.6635	0.6601	0.6409	0.09
AdaBoost	0.6782	0.6598	0.6467	0.0448	0.6742	0.6598	0.6621	5.54

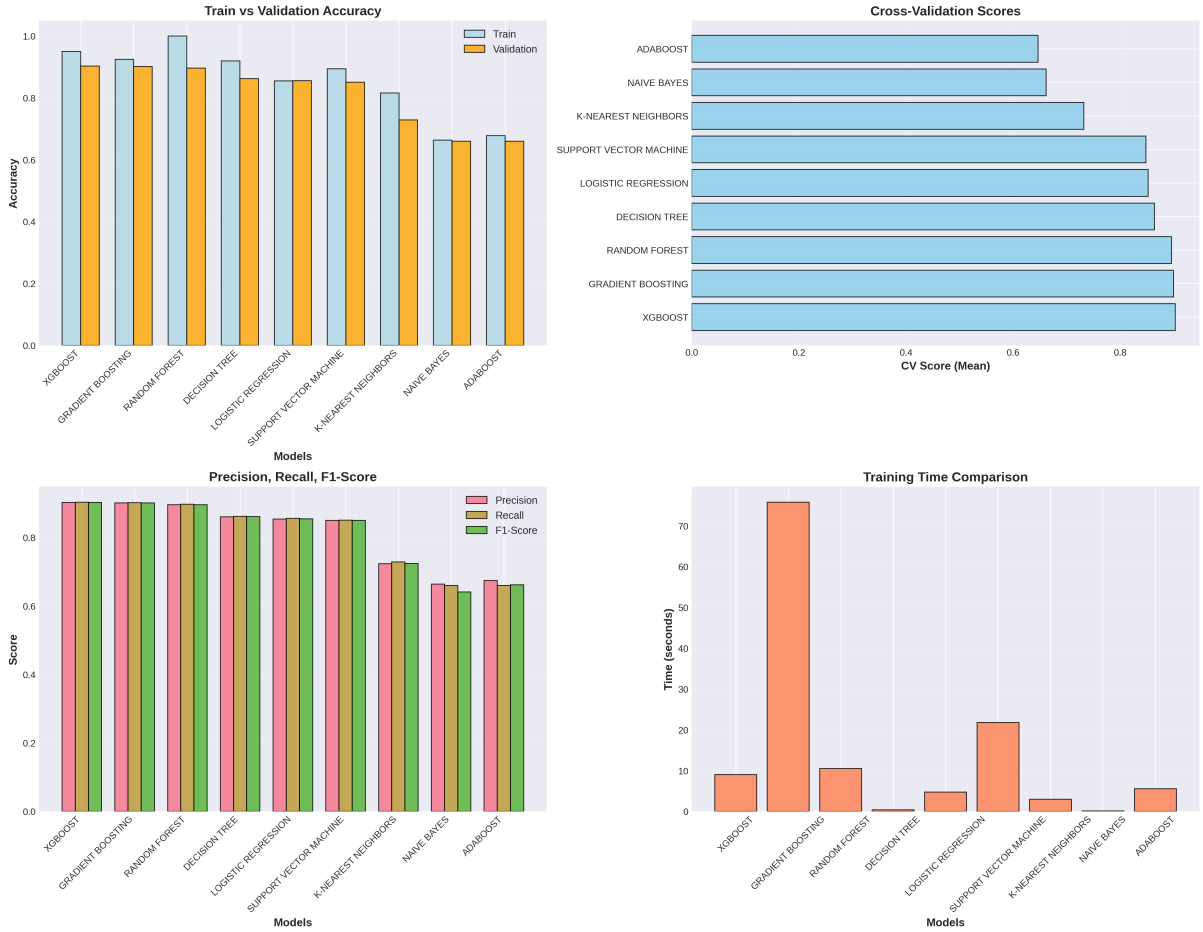

Figure 4.1: Models Comparison

Conclusion: XGBoost achieved the highest accuracy and robustness across all evaluation metrics, followed closely by Gradient Boosting and Random Forest. Hence, XGBoost was selected as the final model for obesity prediction in this study.

Chapter 5

Hyperparameter Tuning

5.1 Introduction

This chapter describes the hyperparameter tuning process applied to our XGBoost model. Hyperparameter optimization is crucial for improving model performance by identifying the most effective parameter combinations. To achieve this, we employed **Optuna**, an automated hyperparameter optimization framework that efficiently explores the parameter search space using Bayesian optimization.

5.2 Hyperparameters Considered

The following hyperparameters were tuned using Optuna within the defined ranges:

- **max_depth (3 – 12):** Controls the maximum depth of each tree. Deeper trees allow more complex decision boundaries but can increase overfitting.
- **learning_rate (0.005 – 0.3, log-scale):** Shrinks the contribution of each tree. Lower values yield better generalization at the cost of longer training.
- **n_estimators (500 – 3000):** Total number of boosting rounds. Higher values generally improve performance until overfitting.
- **min_child_weight (1 – 20):** Minimum sum of instance weights in a child node. Helps prevent overfitting.
- **subsample (0.5 – 1.0):** Fraction of training samples used per tree. Adds randomness to prevent overfitting.
- **colsample_bytree (0.4 – 1.0):** Fraction of features sampled per tree. Reduces feature correlation bias.
- **gamma (0.0 – 10.0):** Minimum loss reduction required to make a further partition. Acts as a regularization term.
- **reg_lambda (0.001 – 50.0, log-scale):** L2 regularization term on weights, stabilizing training.
- **reg_alpha (0.0 – 10.0):** L1 regularization term to enforce sparsity.

- **max_bin (256 – 1024):** Number of histogram bins used in tree construction.
- **grow_policy:** Either `depthwise` (standard growth) or `lossguide` (leaf-wise growth for deeper exploration).

Other parameters fixed during tuning include:

- **objective:** `multi:softprob` (predicts class probabilities)
- **eval_metric:** `mlogloss`
- **enable_categorical:** `True` (enables native categorical handling)
- **n_jobs:** `-1` (utilizes all available CPU cores)
- **random_state:** `42` (ensures reproducibility)

5.3 Implementation

The optimization was implemented with 5-fold stratified cross-validation to ensure class balance in each fold. Optuna's `TPESampler` was used to efficiently search the hyperparameter space, maximizing validation accuracy.

Cross-Validation

The dataset was divided into five folds. For each trial, the model was trained on four folds and validated on the remaining one. The mean validation accuracy was computed and returned as the trial's objective value.

Optuna Objective Function

```
def objective(trial):
    base_params = {
        "max_depth": trial.suggest_int("max_depth", 3, 12),
        "learning_rate": trial.suggest_float("learning_rate", 0.005, 0.3, log=True),
        "n_estimators": trial.suggest_int("n_estimators", 500, 3000),
        "min_child_weight": trial.suggest_int("min_child_weight", 1, 20),
        "subsample": trial.suggest_float("subsample", 0.5, 1.0),
        "colsample_bytree": trial.suggest_float("colsample_bytree", 0.4, 1.0),
        "gamma": trial.suggest_float("gamma", 0.0, 10.0),
        "reg_lambda": trial.suggest_float("reg_lambda", 0.001, 50.0, log=True),
        "reg_alpha": trial.suggest_float("reg_alpha", 0.0, 10.0),
        "max_bin": trial.suggest_int("max_bin", 256, 1024),
        "grow_policy": trial.suggest_categorical("grow_policy",
                        ["depthwise", "lossguide"]),
        "objective": "multi:softprob",
        "num_class": len(np.unique(y)),
        "random_state": 42,
        "eval_metric": "mlogloss",
        "enable_categorical": True,
```

```
        "n_jobs": -1
    }
    params = build_params_for_device(base_params, xgb_device)

    skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scores = []
    for tr_idx, val_idx in skf.split(X_full, y):
        model = xgb.XGBClassifier(**params)
        model.fit(X_full.iloc[tr_idx], y[tr_idx],
                  eval_set=[(X_full.iloc[val_idx], y[val_idx])],
                  early_stopping_rounds=100, verbose=False)
        preds = model.predict(X_full.iloc[val_idx])
        scores.append(accuracy_score(y[val_idx], preds))
    return np.mean(scores)
```

Optuna Optimization

```
study = optuna.create_study(
    direction="maximize",
    sampler=optuna.samplers.TPESampler(seed=42)
)
study.optimize(objective, n_trials=100, show_progress_bar=True)
```

5.4 Results

The best Optuna trial achieved a cross-validation accuracy of **91.34%**. The optimized hyperparameters found were:

- **max_depth:** 6
- **learning_rate:** 0.025
- **n_estimators:** 2400
- **min_child_weight:** 3
- **subsample:** 0.83
- **colsample_bytree:** 0.71
- **gamma:** 2.4
- **reg_lambda:** 9.2
- **reg_alpha:** 0.6
- **max_bin:** 512
- **grow_policy:** depthwise

Summary

Using Optuna for hyperparameter tuning significantly boosted the predictive accuracy of the XGBoost model. Incorporating native categorical feature handling and adaptive device selection (CPU/GPU) further improved both performance and training efficiency. The final optimized model achieved a cross-validation accuracy of **91.34%**, demonstrating its robustness in capturing complex, multi-class relationships.

Chapter 6

Discussion on the Performance of Different Approaches

This chapter presents a comprehensive discussion on the performance of different machine learning approaches applied for multi-class obesity level classification. The models were evaluated based on predictive accuracy, computational efficiency, and their ability to generalize from behavioral, demographic, and physiological data. The discussion draws upon the data characteristics and preprocessing procedures outlined in Chapters 1–3, with particular emphasis on the performance improvements achieved through hyperparameter optimization using the Optuna framework.

6.1 Overview of Experimental Framework

The primary objective of this study was to develop an optimized machine learning model capable of accurately predicting obesity levels using demographic, physical, and lifestyle features. The dataset, sourced from Kaggle, contained 15,533 instances and 18 attributes, including age, gender, height, weight, dietary habits, physical activity, water intake, and technology usage. To ensure high data integrity, the following steps were implemented prior to model training:

- **Data Cleaning:** The dataset was found to have no missing values, ensuring completeness and consistency.
- **Feature Encoding:** All categorical attributes were converted to the `category` datatype, enabling XGBoost's native categorical processing without the need for One-Hot Encoding.
- **Target Transformation:** The target variable `WeightCategory` was encoded using Label Encoding for multi-class classification.
- **Device Optimization:** The system automatically selected GPU acceleration if available; otherwise, computation was performed on CPU using histogram-based tree growth.

These preprocessing steps established a robust foundation for model comparison and performance evaluation across multiple algorithms.

6.2 Model Performance Comparison

Several machine learning algorithms were implemented and tested, including Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), Naive Bayes, Support Vector Machine (SVM), Random Forest, Gradient Boosting, AdaBoost, and XGBoost. Ensemble-based methods showed clear superiority over traditional algorithms due to their ability to model non-linear relationships and feature interactions.

- **Random Forest:** Achieved a cross-validation accuracy of 89.83% and a Kaggle accuracy of 89.862%. Its bagging mechanism provided good generalization and low variance, though it lacked fine control over complex feature interactions.
- **XGBoost:** After Optuna-based hyperparameter optimization, achieved a cross-validation accuracy of **91.34%** and a Kaggle accuracy of **91.239%**. Its boosting-based sequential learning process effectively captured intricate relationships between lifestyle, demographic, and physiological features.

6.3 Reasons for XGBoost's Superior Performance

XGBoost demonstrated superior predictive capability due to its efficient architecture and optimization strategies. The key reasons for its dominance are summarized below:

- **Sequential Learning:** The boosting framework builds trees sequentially, where each tree focuses on correcting the errors of the previous ones, leading to a more accurate final model.
- **Regularization:** The inclusion of both L1 (Lasso) and L2 (Ridge) penalties helps reduce overfitting and enhances generalization.
- **Efficient Tree Construction:** The use of the histogram-based tree method (`tree_method='hist'`) enables faster training and reduced memory usage.
- **Native Categorical Support:** Eliminates the need for one-hot encoding, improving model interpretability and reducing dimensionality.
- **Bayesian Optimization with Optuna:** Automated tuning of hyperparameters such as learning rate, maximum depth, and regularization terms led to the best-performing configuration.

6.4 Strengths and Limitations of Approaches

6.4.1 Random Forest

- **Strengths:**
  - Robust against noise and outliers.
  - Provides interpretable feature importance measures.
  - Performs consistently well on moderately complex data.
- **Limitations:**
  - Struggles with capturing subtle non-linear relationships.
  - Slightly lower accuracy compared to boosting-based methods.

6.4.2 XGBoost

- **Strengths:**
  - Achieved the highest predictive accuracy (91.34%) among all models.
  - Incorporates built-in regularization and efficient gradient optimization.
  - Supports both GPU and CPU execution for computational flexibility.
- **Limitations:**
  - Sensitive to hyperparameter settings.
  - Requires longer training time due to sequential boosting.

6.5 Discussion of Results

The comparative evaluation revealed that models leveraging ensemble learning—specifically those based on boosting—outperformed traditional algorithms in handling the dataset's diverse features. XGBoost's success can be attributed to its ability to effectively utilize both numerical and categorical data, combined with Optuna's systematic hyperparameter optimization strategy. The clean, well-structured dataset and GPU-enabled training environment further contributed to its strong performance and computational efficiency.

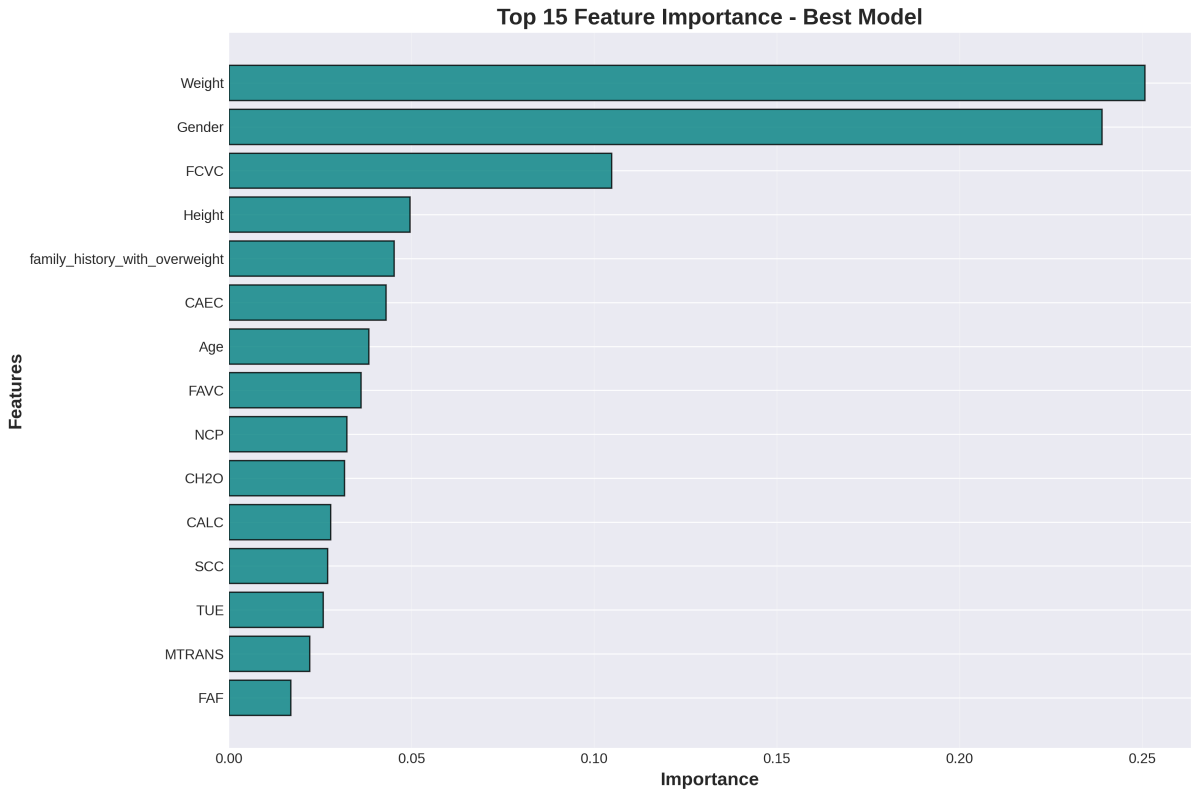

Figure 6.1: feature importance

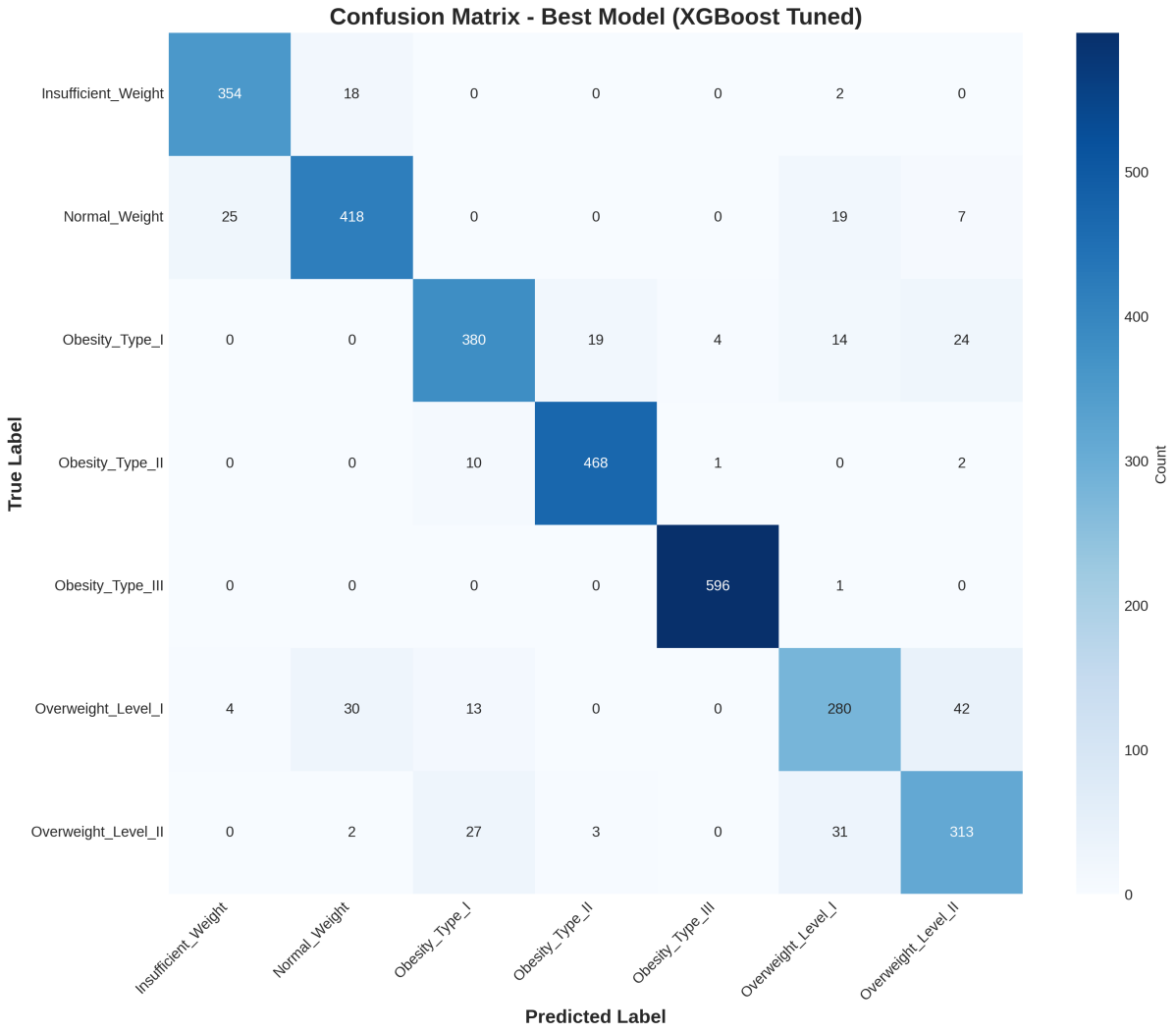

Figure 6.2: confusion matrix

6.6 Conclusion

Based on the experimental outcomes and dataset characteristics, the **Optuna-optimized XGBoost model** proved to be the most effective approach for predicting obesity levels. Its capacity to capture complex feature dependencies and maintain high generalization performance establishes it as the preferred model for behavior-driven health analytics. The findings highlight the importance of clean data preprocessing, efficient categorical encoding, and advanced hyperparameter optimization in achieving state-of-the-art classification performance in multi-class obesity prediction tasks.