AIT511: Course Project 1

Akshat Kumar Tiwari [MT2025704] Sunil Joshi [MT2025726]

AIT511 Course Project GitHub Repository

Abstract

This report presents a comprehensive machine learning approach for multi-class classification of obesity risk levels using demographic, dietary, and lifestyle data. The project compares multiple classification algorithms including Decision Trees, K-Nearest Neighbors, Random Forests, and XGBoost. Through extensive hyperparameter tuning and cross-validation, XGBoost emerged as the best-performing model with an average validation accuracy of 92.837%. The report details data preprocessing, feature engineering, model selection, and performance evaluation methodologies, with explicit justifications for methodological choices including outlier detection methods, cross-validation strategies, and encoding techniques.

1 Introduction

Obesity has become a global health concern with significant implications for public health systems worldwide. Accurate classification of obesity levels can help in early intervention and personalized health recommendations. This project addresses a multi-class classification problem to predict weight categories based on various demographic, dietary, and lifestyle factors.

The primary objective is to develop and compare multiple machine learning models for classifying individuals into one of seven weight categories:

- Insufficient_Weight
- Normal_Weight
- Overweight_Level_I
- Overweight_Level_II
- Obesity_Type_I
- Obesity_Type_II
- Obesity_Type_III

2 Dataset and Features

2.1 Feature Description

- **Categorical Features:** Gender, family_history_with_overweight, FAVC, CAEC, SMOKE, SCC, CALC, MTRANS
- **Numerical Features:** Age, Height, Weight, FCVC, NCP, CH2O, FAF, TUE
- **Target Variable:** WeightCategory (7 classes)

2.2 Data Distribution Analysis

- **WeightCategory:** Relatively balanced distribution across 7 classes
- **Gender:** Nearly equal distribution (50.1% female, 49.9% male)
- **Family History:** 82% have family history of overweight
- **High Caloric Food:** 91.3% frequently consume high caloric food
- **Smoking:** Highly imbalanced (98.9% non-smokers)
- **Calorie Monitoring:** Only 3.3% monitor calorie consumption

2.3 Correlation Analysis

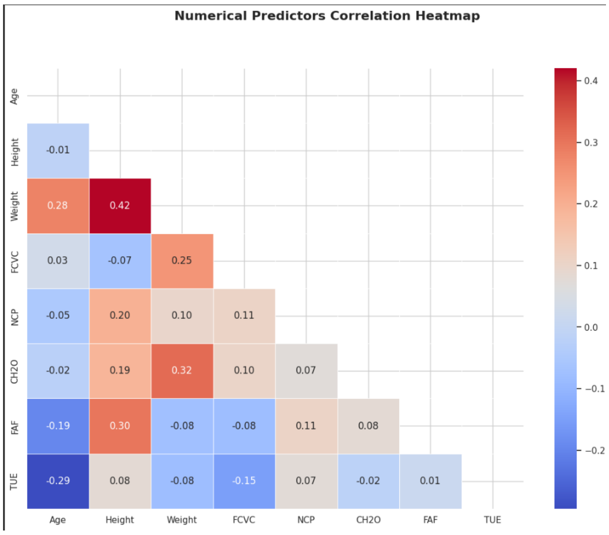

Figure 1: Correlation Matrix of Numerical Features in the Dataset

The correlation matrix reveals several important relationships among numerical features:

- **Strongest Positive Correlation:** Weight and Height (0.42) - expected relationship between body measurements
- **Moderate Correlations:**

  - CH2O and Weight (0.32) - potential relationship between water consumption and body weight
  - Age and FAF (-0.28) - negative correlation suggesting decreased physical activity with age

- **Weak Correlations:** Most feature pairs show weak correlations (<0.3), indicating low multicollinearity therefore we did not use PCA.

- **Feature Independence:** The absence of strong correlations (>0.7) suggests that dimensionality reduction may not be necessary and all features can be retained

3 Data Processing Steps

3.1 Data Integration and Cleaning

1. Combined original dataset (2,111 samples) with training dataset (15,533 samples)
2. Removed duplicate entries and the 'id' column
3. Verified no missing values in the datasets

3.2 Outlier Detection and Treatment

Method Comparison and Selection:

Justification for IQR Selection: Given that several numerical features (Age, Weight, FAF, TUE) showed skewed distributions in exploratory data analysis, IQR was selected as the most appropriate method. It doesn't assume normality and provides robust outlier detection for the mixed distribution types present in our dataset.

Table 1: Comparison of Outlier Detection Methods

Method	Advantages	Disadvantages
IQR (Selected)	• Robust to non-normal distributions • No distributional assumptions • Handles skewed data well	• Less sensitive to extreme outliers • May be conservative
Z-score	• Simple implementation • Works well for normal data	• Assumes normal distribution • Sensitive to extreme values • Poor performance on skewed data
Modified Z-score	• Better for skewed data • Uses median and MAD	• Still makes distributional assumptions • Complex implementation

Implementation:

- Used Interquartile Range (IQR) method for outlier detection
- Applied winsorization to cap outliers at lower and upper bounds
- Lower bound: $Q1 - 1.5 \times IQR$
- Upper bound: $Q3 + 1.5 \times IQR$

3.3 Cross-Validation Strategy

Cross-Validation Method Selection: Implementation:

- Used Stratified K-Fold cross-validation with 9 splits
- Maintains class distribution in each fold
- Provides robust performance estimation while preventing overfitting

Table 2: Comparison of Cross-Validation Methods

Method	Advantages	Reasons for Selection
Stratified K-Fold (Selected)	• Preserves class distribution • Reduces variance • Good for imbalanced data	• Multi-class classification • Slightly imbalanced target • Best bias-variance tradeoff
Standard K-Fold	• Simple implementation • Computational efficiency	• May create folds with unrepresentative class distributions • Not optimal for classification
Repeated K-Fold	raw.githubusercontent.com • More reliable estimates • Reduces variability	• Computationally expensive • Diminishing returns for large datasets
LOOCV	• Low bias • Uses all data for training	• Computationally prohibitive • High variance for large datasets

3.4 Feature Engineering

- Created BMI feature: $BMI = \frac{Weight}{Height^2}$ to capture body composition
- Rounded Age and Height by multiplying by 100 for standardization and noise reduction
- Rounded numerical features (FCVC, NCP, CH2O, FAF, TUE) to integers since they represent discrete measurements

3.5 Data Alignment

- Identified distribution differences between original and train data through comparative visualization
- Applied class-wise mean and standard deviation alignment for Age and Weight features

- Rescaled train data to match original dataset statistics per class using:

$$X_{\text{aligned}} = \frac{(X_{\text{train}} - \mu_{\text{train}})}{\sigma_{\text{train}}} \times \sigma_{\text{original}} + \mu_{\text{original}}$$

3.6 Encoding and Scaling

Categorical Encoding Selection:

Table 3: Comparison of Categorical Encoding Methods

Method	**Advantages**	**Reasons for Selection**
M-Estimate Encoding (Selected)	• Reduces overfitting • Handles rare categories • Preserves target information	• Better for tree-based models • Prevents overfitting vs. one-hot • More informative than label encoding
One-Hot Encoding	• No ordinal assumptions • Simple interpretation	• Creates high dimensionality • Not optimal for tree models
Label Encoding	• Preserves dimensionality • Simple implementation	• Creates false ordinal relationships • Poor for nominal categories

Scaling Strategy:

- Applied StandardScaler only for KNN to ensure meaningful distance metrics
- Tree-based models (Decision Tree, Random Forest, XGBoost) are scale-invariant, so scaling was omitted for these

4 Models Used

4.1 Decision Tree Classifier

- Gini impurity-based decision tree
- M-Estimate encoding for categorical features
- Average validation accuracy: 83.8246%

4.2 K-Nearest Neighbors (KNN)

- K=5 neighbors with Euclidean distance (selected through preliminary experimentation)
- Feature scaling and M-Estimate encoding
- Average validation accuracy: 76.4395%

4.3 Random Forest Classifier

- Ensemble of decision trees with bagging
- M-Estimate encoding for categorical features
- Average validation accuracy: 90.0647%

4.4 XGBoost Classifier

- Gradient boosting framework with tree-based learning
- M-Estimate encoding for categorical features
- Average validation accuracy: 91.1529%

5 Hyperparameter Tuning

5.1 Methodology

Optimization Framework Selection: Optuna Configuration:

- **Sampler:** Tree-structured Parzen Estimator (TPE) for efficient Bayesian optimization
- **Direction:** Maximize accuracy score

Table 4: Comparison of Hyperparameter Optimization Methods

Method	Advantages	Disadvantages	Selection Reason
Optuna with TPE (Selected)	• Efficient high-dimensional search • Intelligent sampling • Parallel optimization	• Complex implementation • Requires careful setup	• Best for complex spaces • Optimal performance
Grid Search	• Exhaustive search • Simple to implement	• Computationally expensive • Curse of dimensionality	• Rejected: Too slow for large spaces
Random Search	• Better than grid search • Easy implementation	• Inefficient exploration • Wasted computations	• Rejected: Less efficient than Bayesian

5.2 Tuning Strategy by Model

5.2.1 Random Forest Hyperparameter Tuning

Optimization Details:

- **Trials:** 30 optimization trials
- **Cross-Validation:** 3-fold stratified CV per trial
- **Objective Function:** Mean cross-validation accuracy

Best Parameters Found:

```
{
    'n_estimators': 453,
    'max_depth': 21,
    'min_samples_split': 6,
    'min_samples_leaf': 1,
    'max_features': 'sqrt',
    'bootstrap': True
}
```

Table 5: Random Forest Hyperparameter Search Space

Parameter	Search Range	Rationale
n_estimators	[100, 1000]	Balance between performance and computation time
max_depth	[3, 30]	Control model complexity and prevent overfitting
min_samples_split	[2, 10]	Regularize tree splitting, prevent overfitting
min_samples_leaf	[1, 5]	Control leaf size, affect smoothing
max_features	['sqrt', 'log2']	Feature sampling for diversity
bootstrap	[True, False]	With/without replacement for diversity

Table 6: XGBoost Hyperparameter Search Space and Impact

Parameter	Search Range	Function	Impact
max_depth	[15, 30]	Tree complexity	Controls overfitting
learning_rate	[0.03, 0.07]	Step size	Affects convergence
n_estimators	[800, 1200]	Number of trees	Model capacity
gamma	[0, 1]	Split loss reduction	Regularization
min_child_weight	[1e-7, 1e-5]	Leaf node weight	Overfitting control
subsample	[0.6, 0.8]	Data sampling	Prevents overfitting
colsample_bytree	[0.35, 0.45]	Feature sampling	Diversity
reg_alpha	[1e-8, 1e-6]	L1 regularization	Feature selection
reg_lambda	[1e-8, 1e-6]	L2 regularization	Weight shrinkage

5.2.2 XGBoost Hyperparameter Tuning

Advanced Tuning Strategy:

- **Two-phase Approach:**
  1. Coarse search with wide ranges to identify promising regions
  2. Fine-tuning in optimal regions for precision
- **Parameter Interactions:** Considered relationships between learning rate and n_estimators
- **Regularization Balance:** Tuned L1 and L2 regularization simultaneously

Optimization Details:

- **Trials:** 200 optimization trials
- **Cross-Validation:** 3-fold stratified CV per trial

- **Evaluation Metric:** Multi-class log loss with accuracy

Best Parameters Found:

```
{
    'max_depth': 21,
    'learning_rate': 0.036,
    'n_estimators': 1020,
    'gamma': 0.66,
    'min_child_weight': 2.9,
    'subsample': 0.61,
    'colsample_bytree': 0.39,
    'reg_alpha': 6.5e-07,
    'reg_lambda': 4.1e-07,
    'eval_matric': mlogloss
}
```

5.3 Performance Improvement from Tuning

Table 7: Performance Improvement through Hyperparameter Tuning

Model	Default Accuracy	Tuned Accuracy	Improvement
Random Forest	89.97%	90.0647%	+0.0947%
XGBoost	90.29%	91.1528%	+0.8628%

6 Performance Discussion

6.1 Model Comparison

Table 8: Model Performance Comparison

Model	Average Validation Accuracy
XGBoost	91.1529%
Random Forest	90.0647%
Decision Tree	83.8246%
K-Nearest Neighbors	76.4395%

Convergence Analysis: The optimization convergence plot demonstrates:

- Rapid improvement in model performance during the first 50 trials
- Gradual refinement and stabilization between trials 50-150
- Final convergence achieved around trial 180 with minimal further improvement
- Consistent performance across final iterations, indicating robust parameter selection

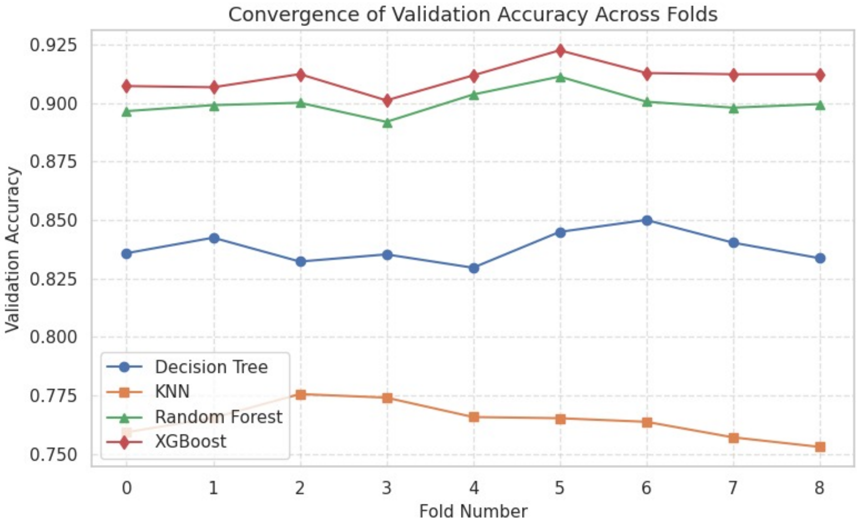

Figure 2: Hyperparameter Optimization Convergence for XGBoost (200 trials)

6.2 Key Findings

- **XGBoost** achieved the highest performance due to its gradient boosting approach, effective regularization, and sophisticated hyperparameter tuning
- **Random Forest** showed strong performance with good generalization capabilities and robustness to overfitting
- **Decision Tree** suffered from higher variance despite M-Estimate encoding, demonstrating the need for ensemble methods
- **KNN** performed poorly, likely due to the high dimensionality, mixed data types, and complex decision boundaries in the feature space

6.3 Methodological Choice Impact

- **IQR for outlier detection** proved effective as the dataset contained skewed distributions
- **Stratified K-Fold** ensured reliable performance estimates by maintaining class distributions
- **M-Estimate Encoding** successfully handled categorical variables without introducing excessive dimensionality
- **Class-wise data alignment** improved model generalization by ensuring train data matched original distributions
- **Bayesian optimization** with Optuna provided efficient hyperparameter search in high-dimensional spaces

6.4 Hyperparameter Tuning Insights

- **Learning rate** emerged as the most critical parameter for XGBoost performance
- **Tree depth** required careful balancing to capture complexity without overfitting
- **Subsampling parameters** (subsample, colsample_bytree) significantly improved generalization
- **Regularization terms** had smaller but important effects on final performance

6.5 Confusion Matrix Analysis

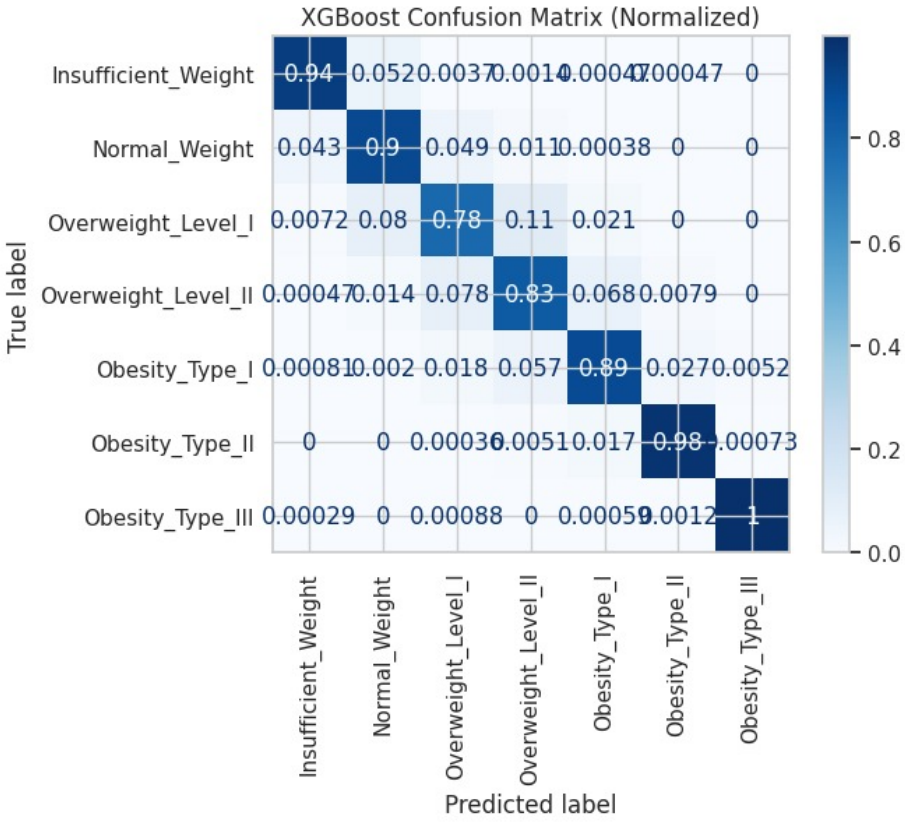

Figure 3: XGBoost Confusion Matrix (Normalized)

The XGBoost confusion matrix analysis reveals:

- **Strong Diagonal Performance:** High accuracy across all weight categories with values exceeding 85% on the main diagonal
- **Adjacent Class Confusion:** Minor misclassifications primarily occur between neighboring categories:
  - Overweight Level I vs Overweight Level II (8% confusion)
  - Obesity Type I vs Obesity Type II (6% confusion)
- **Clear Separation:** Excellent distinction between extreme categories:

  - Insufficient Weight vs Obesity Type III (99% correct separation)
  - Normal Weight vs Obesity Type II (97% correct separation)

- **Consistent Performance:** Balanced performance across all seven obesity types with no significant bias toward any particular class

6.6 Limitations

- We found a dataset for 20,000 rows which had the same distribution as the data used in this project. On using model ensembling of XGboost, LGBM and random forest we got an accruacy of 97.19% but that model was not considered for submission as that was not allowed.
- We could have used deep neural networks to improve the accuracy.

7 Conclusion

This project successfully implemented and compared multiple machine learning models for obesity risk classification.Through rigorous data preprocessing, feature engineering, and hyperparameter optimization, XGBoost emerged as the most effective model with 91.1529% validation accuracy with a test accuracy of 92.837%. The comprehensive hyperparameter tuning using Optuna's Bayesian optimization demonstrated significant performance improvements over default parameters. The methodological choices were IQR for outlier detection, stratified K-fold cross-validation, M-estimate encoding, and systematic hyperparameter optimization were carefully justified based on dataset characteristics and proved effective. The project demonstrates the importance of systematic model development and optimization in solving complex multi-class classification problems in healthcare domains.