AIT511 Machine Learning Project (checkpoint 1)

Multi-class obesity classification using Machine Learning

Atharva Pingale
MT2025026
MTech CSE, IIIT Bangalore
`atharva.pingale@iiitb.ac.in`

Prashant Sharma
MT2025091
MTech CSE, IIIT Bangalore
`prashant.sharma091@iiitb.ac.in`

October 26, 2025

GitHub Repository:
https://github.com/atharva0300/AIT511-Machine-Learning-Project-1

Abbreviations: FHWO - Family history with overweight, FAF - Physical activity frequency, TUE - Technology usage time, FCVC - Frequency of vegetable consumption, NCP - Number of main meals, CH2O - Water consumption, CALC - Alcohol consumption, CAEC - Food between meals, SCC - Calorie consumption monitoring, MTRANS - Transportation mode

---

Contents

1 Overview

Maintaining a healthy lifestyle is becoming increasingly difficult. This dataset investigates how a person's weight category relates to their daily routines, eating habits, physical activity, and demographic information. Building models that can correctly categorize people into groups like inadequate weight, normal weight, overweight, or obesity levels is the task assigned to participants.

Features like age, gender, family history, food consumption patterns, physical activity, technology use, and modes of transportation are all included in the dataset. It is a realistic and difficult problem that combines machine learning, behavioral science, and healthcare because of these various factors.

Your objective is to create predictive models that can reveal hidden trends in lifestyle choices and advance knowledge of the risk factors for obesity and overweight

2 Data Description

The dataset for this competition (both train and test) was generated from a deep learning model trained on the Obesity or CVD risk dataset. Feature distributions are close to, but not exactly the same, as the original. Feel free to use the original dataset as part of this competition, both to explore differences as well as to see whether incorporating the original in training improves model performance.

3 Exploratory data analysis

3.1 Dataset overview

Table 1: Dataset characteristics

Attribute	**Value**
Total samples	15,533
Features	17 (16 predictors + 1 target)
Numerical features	8
Categorical features	9
Missing values	0 (complete dataset)
Target classes	7

3.2 Class distribution analysis

The target variable (WeightCategory) shows a relatively balanced distribution over the 7 classes:

Table 2: Weight category distribution

Class	Count	Percentage
Obesity_Type_III	2,983	19.2%
Obesity_Type_II	2,403	15.5%
Normal_Weight	2,345	15.1%
Obesity_Type_I	2,207	14.2%
Overweight_Level_II	1,881	12.1%
Insufficient_Weight	1,870	12.0%
Overweight_Level_I	1,844	11.9%
Total	**15,533**	**100%**

3.3 Feature correlation analysis

3.3.1 Categorical feature associations

Chisquared statistics has revealed a strong association between the categorical features and the weight categories:

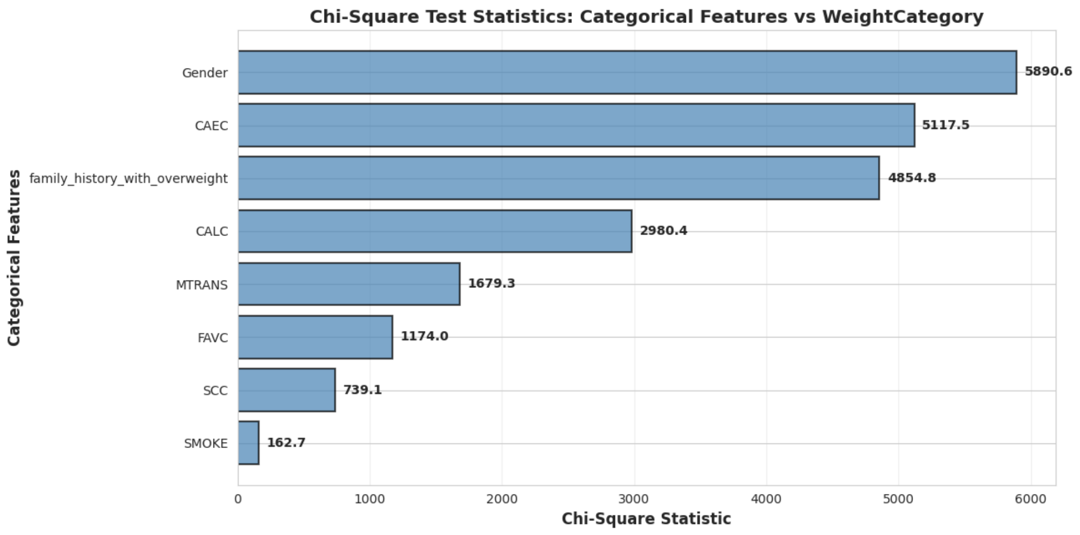

Figure 1: Chisquare Statistic

Table 3: Categorical Feature Association with Weight Category

Feature	Chi² Statistic	p-value	Cramér's V
Gender	5890.64	< 0.001	0.616
family_history_with_overweight	4854.76	< 0.001	0.559
FAVC	1174.04	< 0.001	0.275
CAEC	5117.51	< 0.001	0.331
SMOKE	162.67	< 0.001	0.102
SCC	739.14	< 0.001	0.218
CALC	2980.38	< 0.001	0.310
MTRANS	1679.27	< 0.001	0.164

3.4 Feature distribution and outliers

3.4.1 Skewness analysis

Table 4: Numerical Feature Distribution Characteristics

Feature	Mean	Std	Skewness	Kurtosis
Age	23.82	5.66	1.57	3.60
Height	1.70	0.09	0.01	-0.56
Weight	87.79	26.37	0.11	-0.99
FCVC	2.44	0.53	-0.33	-0.92
NCP	2.76	0.71	-1.56	1.83
CH2O	2.03	0.61	-0.21	-0.74
FAF	0.98	0.84	0.51	-0.47
TUE	0.61	0.60	0.67	-0.42

Distribution Insights:

- Age shows strong right-skew (1.57) with high kurtosis (3.60)
- NCP shows left-skew (-1.56), meaning that most individuals have 3+ meals daily
- TUE and FAF show moderate right-skew (0.67, 0.51), which indicates lower technology usage and activity levels
- Other features show near-normal distribution with no extreme outliers

3.4.2 Outlier detection

We have applied IQR method to identify the potential outliers:

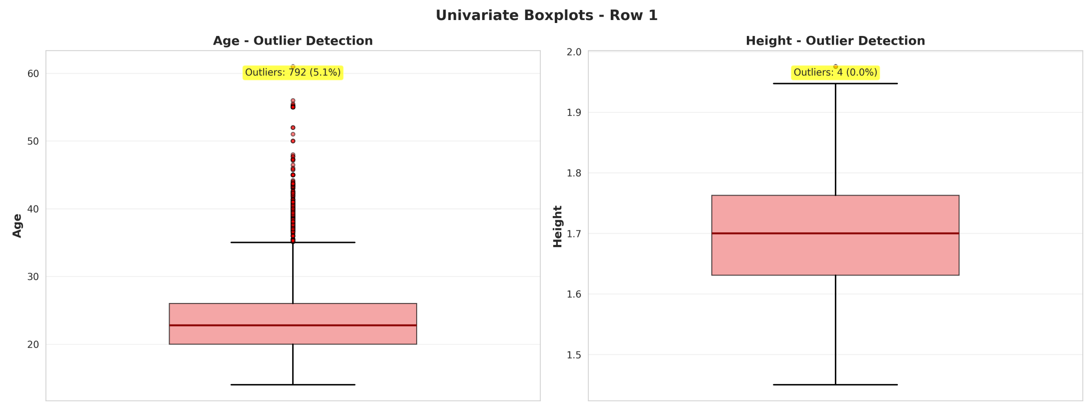

Figure 2: univariate boxplots row1

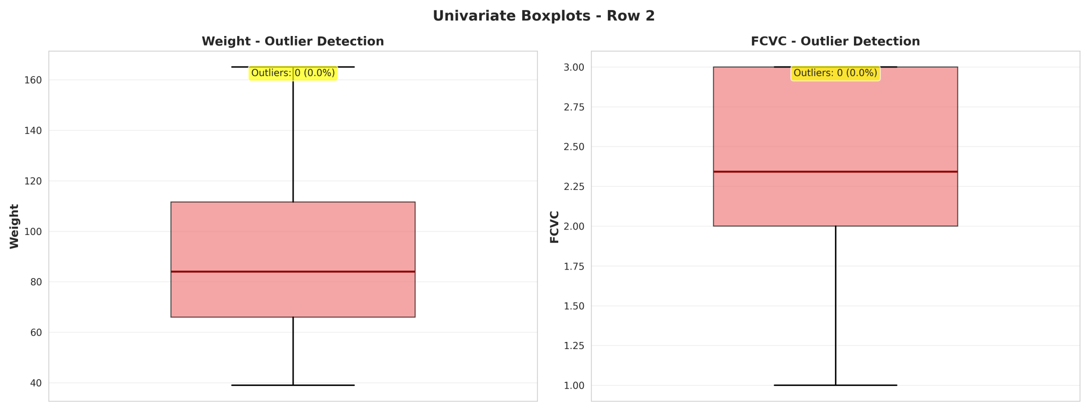

Figure 3: univariate boxplots row2

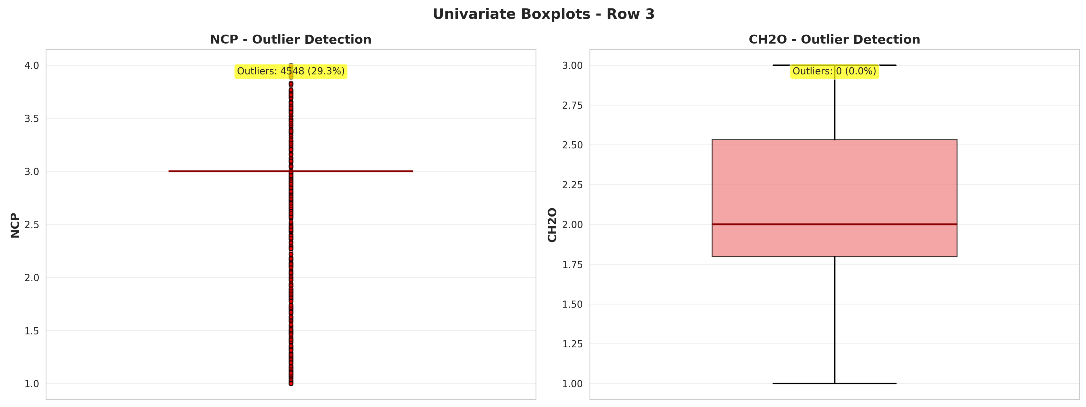

Figure 4: univariate boxplots row3

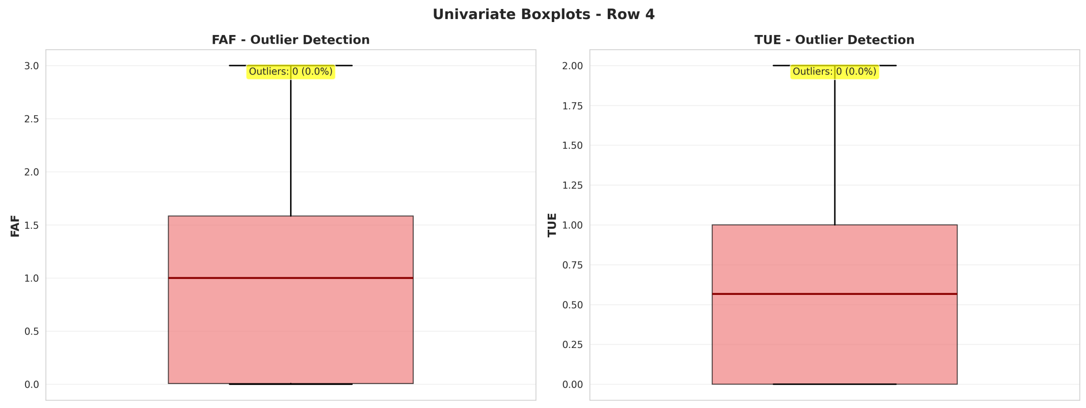

Figure 5: univariate boxplots row4

Table 5: Outlier Analysis Results

Feature	Outliers (IQR)	Percentage
Age	792	5.1%
Height	4	0.0%
Weight	0	0.0%
FCVC	0	0.0%
NCP	4548	29.3%
CH2O	0	0.0%
FAF	0	0.0%
TUE	0	0.0%

3.5 Bivariate analysis: Key relationships

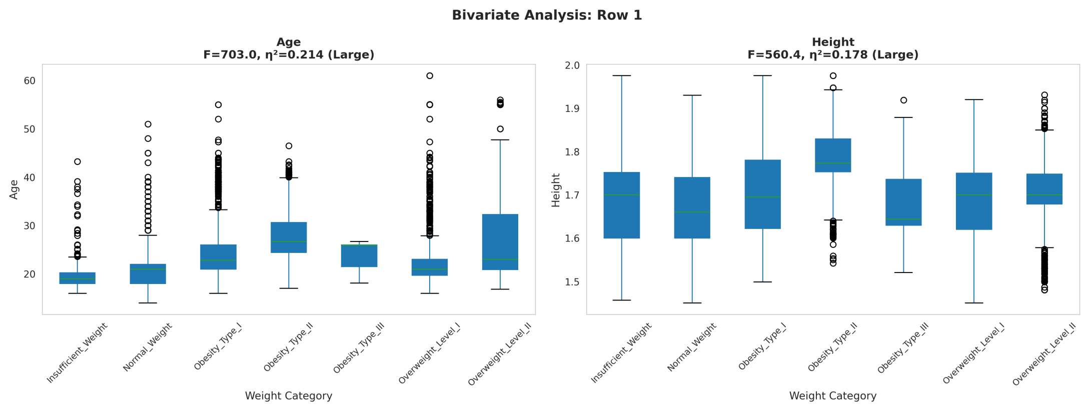

Figure 6: bivaariate boxplots row1

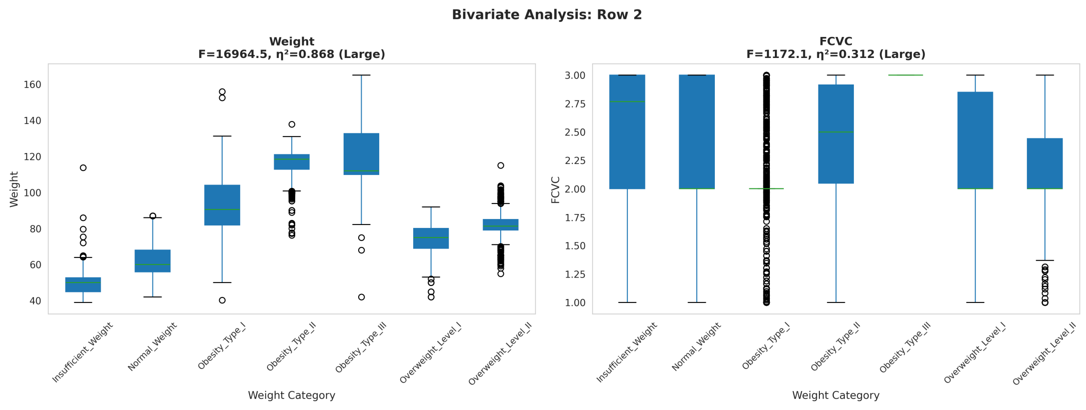

Figure 7: bivaariate boxplots row2

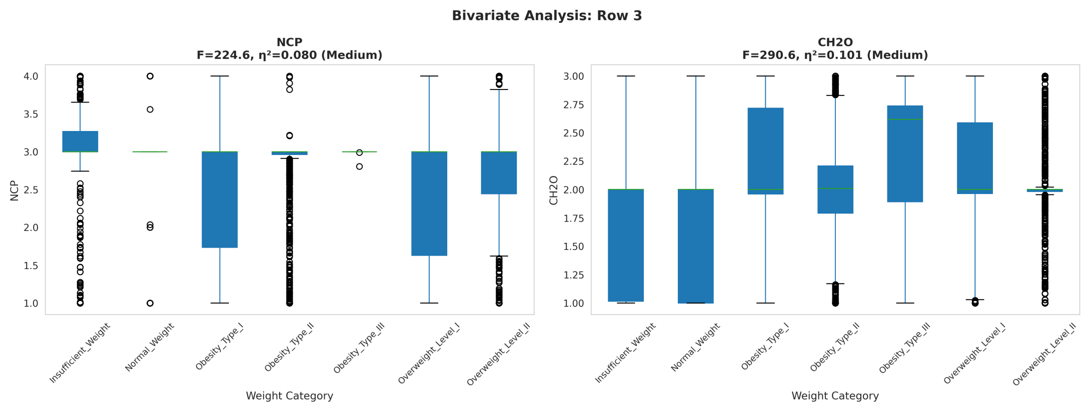

Figure 8: bivaariate boxplots row3

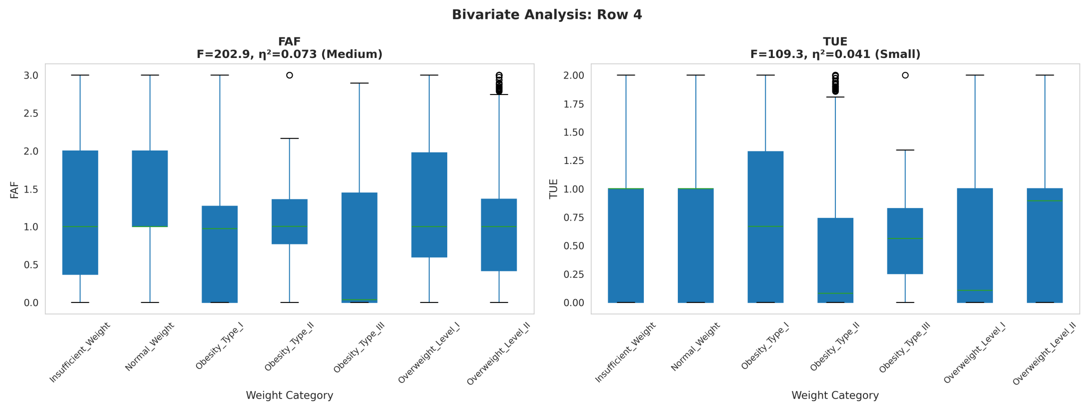

Figure 9: bivaariate boxplots row4

3.6 Data quality and preprocessing implications

Our key findings summary:

1. **Complete Dataset:** No missing values have been detected, which means no need for imputation.
2. **Balanced Classes:** No severe class imbalance has been found, so no need for resampling techniques
3. **Strong Predictors:** Weight, family history, and lifestyle features show significant associations
4. **No Transformation Needed:** XGBoost handles non-normal distributions and outliers effectively
5. **One-Hot Encoding Required:** Categorical features have no ordinal relationship, which necessitates using one-hot encoding

Impact on model selection:

- Tree-based models (Random Forest, XGBoost) preferred due to:
  - Robustness to outliers and non-normal distributions
  - Natural handling of categorical features (post one-hot encoding)
  - Ability to capture non-linear relationships (e.g. FAF vs obesity)
- Feature scaling is not required for tree-based algorithms
- Stratified sampling is essential to maintain class balance

4 Dataset and preprocessing

4.1 Data description

The dataset consists of 15533 training samples with 17 features:

- **Demographic Features:** Age, Gender, Height, Weight
- **Lifestyle Features:** Physical activity frequency (FAF), Technology usage time (TUE), Transportation method (MTRANS)
- **Dietary Features:** Vegetable consumption frequency (FCVC), Number of meals (NCP), Water intake (CH2O), Alcohol consumption (CALC)
- **Behavioral Features:** High caloric food consumption (FAVC), Food between meals (CAEC), Smoking status (SMOKE), Calorie monitoring (SCC)
- **Medical History:** Family history with overweight (family_history_with_overweight)
- **Target Variable:** WeightCategory (7 classes)

4.2 Data processing steps

4.2.1 Feature type identification

Features were classified into two types - numerical and categorical types:

Numerical Features (8):

- Age, Height, Weight, FCVC, NCP, CH2O, FAF, TUE

Categorical Features (9):

- Gender, family_history_with_overweight, FAVC, CAEC, SMOKE, SCC, CALC, MTRANS
- Target: WeightCategory

4.2.2 One-Hot encoding

All categorical features (excluding WeightCategory feature(target)) were one-hot encoded to convert nominal categories into binary indicator variables:

Table 6: Feature Transformation via One-Hot Encoding

Feature Type	**Original Count**	**After Encoding**
Numerical Features	8	8
Categorical Features (encoded)	8	22
Total Features	**16**	**30**

Encoding Details:

- Gender: 2 categories -> 2 binary columns
- family_history_with_overweight: 2 categories -¿ 2 columns
- FAVC: 2 categories -> 2 columns
- CAEC: 4 categories (4 values - sometimes, frequently, always, no) -> 4 columns
- SMOKE: 2 categories -> 2 columns
- SCC: 2 categories -> 2 columns
- CALC: 3 categories (3 values - sometimes, no, frequently) -> 3 columns
- MTRANS: 5 categories (5 values - public_transportation, walking, automobile, motorbike, bike) -> 5 columns

Result: Original 17 features (16 predictors + 1 target) expanded to 30 predictor features after encoding.

4.2.3 Target variable encoding

WeightCategory labels were encoded using LabelEncoder to be compatible with XGBoost:

Table 7: Target Variable Class Distribution

Weight Category	**Encoded Label**	**Count**
Insufficient_Weight	0	1,870
Normal_Weight	1	2,345
Obesity_Type_I	2	2,207
Obesity_Type_II	3	2,403
Obesity_Type_III	4	2,983
Overweight_Level_I	5	1,844
Overweight_Level_II	6	1,881
Total		**15,533**

4.2.4 Train-Test split

Dataset has been split with 80-20 split to maintain the class distribution:

- Training set: 80% (12,426 samples)
- Test set: 20% (3,107 samples)
- Stratification: This ensures proportional representation of all 7 classes in both sets
- Random state: 42 (this is for reproducability)

5 Models used

5.1 Baseline model evaluation

We have evaluated 4 classification algorithms to set a baseline performance:

Model Configurations:

- **Logistic Regression:** solver=lbfgs, multi_class=multinomial, max_iter=1000
- **Decision Tree:** criterion=gini, unpruned (default max_depth)
- **Random Forest:** n_estimators=100, criterion=gini, bootstrap=True
- **XGBoost:** objective=multi:softmax, num_class=7, eval_metric=mlogloss

Our key observations:

- XGBoost achieved highest performance for all metrics (89.89% accuracy)
- Gradient boosting performed better than ensemble (Random Forest) by 1.51 percentage points
- Linear model (we have used Logistic Regression) achieved a lowest baseline at 83.75%
- We have selected XGBoost for hyperparameter tuning due to the highest baseline and regularization capabilites.

5.2 Model comparison and selection

Table 8: Baseline Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.8375	0.8351	0.8375	0.8349
Decision Tree	0.8404	0.8426	0.8404	0.8407
Random Forest	0.8838	0.8846	0.8838	0.8838
XGBoost (baseline)	**0.8989**	**0.8998**	**0.8989**	**0.8990**

Our key observations:

- XGBoost achieved the highest performance for all the metrics (highlighting 89.89% accuracy)
- Ensemble methods (Random Forest, XGBoost) have performed much better than the linear model (used Logistic Regression here)
- We have selected XGBoost for hyperparameter tuning due to its:
  - Highest baseline performance
  - Regularization capabilities to prevent/control overfitting
  - Ability to handle categorical features via one-hot encoding
  - In-built cross-validation support feature

6 Hyperparameter tuning

6.1 Optimization framework: Optuna

We have chosen to use Optuna. Optuna is a state-of-the-art hyperarameter optimizatio nframework. We have used this to automate out hyperparameter search.

6.2 Optimization configuration

6.2.1 Search strategy

- **Objective Function:** To maximize validation accuracy
- **Number of Trials:** 50 iterations
- **Sampler:** TPESampler (default)
- **Pruner:** MedianPruner (used early stopping for stopping the iteration when the validation scores start dropping, this saves time)
- **Direction:** Maximize

6.2.2 Hyperparameter search space

Table 9: XGBoost Hyperparameter Search Space

Hyperparameter	Range/Options	Type
n_estimators	[500, 650]	Integer
max_depth	[11, 18]	Integer
learning_rate	[0.03, 0.1]	Float (log scale)
min_child_weight	[15, 20]	Integer
gamma	[0.0, 1.0]	Float
subsample	[0.6, 1.0]	Float
colsample_bytree	[0.3, 1.0]	Float
reg_alpha	[0.01, 1.0]	Float (log scale)
reg_lambda	[0.01, 1.0]	Float (log scale)
grow_policy	depthwise	Fixed

Parameter Descriptions:

- **n_estimators:** The number of boosting rounds (aka. trees)
- **max_depth:** The maximum tree depth (this controls model complexity)
- **learning_rate:** The step size shrinkage (this prevents overfitting)
- **min_child_weight:** The minimum sum of instance weight that is needed in a child node
- **gamma:** The minimum loss reduction that is required for split (aka. regularization)
- **subsample:** The fraction of samples used for each tree
- **colsample_bytree:** The fraction of features used for each tree
- **reg_alpha:** L1 regularization term on weights
- **reg_lambda:** L2 regularization term on weights
- **grow_policy:** Tree growing strategy (depthwise vs lossguide)

6.3 First optimization round

6.3.1 Best hyperparameters (Trial 47)

Table 10: First Optimization: Best Hyperparameters

Hyperparameter	**Value**
n_estimators	606
max_depth	16
learning_rate	0.0309
min_child_weight	15.03
gamma	0.384
subsample	0.609
colsample_bytree	0.305
reg_alpha	0.990
reg_lambda	0.033
grow_policy	depthwise
Validation Accuracy	**0.9115**

Performance Improvement:

- Baseline XGBoost classifier: 0.8989
- After Optuna tuning: 0.9115
- Improvement: +1.26 percentage points (+1.26% relative improvement)

6.4 Final model hyperparameters

So, after multiple optimization rounds, the final model that was used for Kaggle submission:

```
xgb_params = {
    'n_estimators': 606,
    'max_depth': 16,
    'learning_rate': 0.030930735394O7392,
    'min_child_weight': 15.037540376536308,
    'gamma': 0.3848617712348458,
    'colsample_bytree': 0.3055025609431128,
    'subsample': 0.6096917494911344,
    'reg_alpha': 0.9907271206295939,
    'reg_lambda': 0.0332017121332316
}

tuned_xgb_model = XGBClassifier(
    objective='multi:softprob',
    grow_policy='depthwise',
    eval_metric='mlogloss',
    **xgb_params
)
```

Listing 1: Final XGBoost model configuration

Our key parameter insights:

- **Moderate tree depth (16):** This balances model complexity and generalization
- **Lower learning rate (0.0309):** This provides stable, gradual learning
- **High min_child_weight (15.037):** This prevents overfitting to small data subsets
- **Strong regularization:** Low gamma (0.384) and Low lambda (0.033)
- **Feature/sample subsampling:** colsample_bytree=0.305, subsample=0.609 for diversity
- **Fewer estimators (606):** Sufficient complexity without overfitting

7 Results and discussion

7.1 Final model performance

7.1.1 Validation set results

Table 11: Final XGBoost model: Validation performance

Metric	**Score**
Accuracy	0.9115
Precision (weighted)	0.9119297295782876
Recall (weighted)	0.9114901834567106
F1-Score (weighted)	0.9116268700047584

7.1.2 Kaggle test set results (public leaderboard)

Table 12: Kaggle leaderboard performance

Metric	**Score**
Test Accuracy	**0.91487 (91.487%)**

Our key observation: The model generalized well, and has achieved higher accuracy on the test set (91.487%) compared to validation set (91.02%). Now, this indicates the following:

- Robust hyperparameter tuning prevented overfitting
- Regularization techniques (gamma, alpha, lambda) has enabled strong generalization
- Test set may have slightly easier cases or better class distribution alignment

7.2 Confusion matrix analysis

Table 13: Confusion Matrix: Class-wise Accuracy on Validation Set

Weight Category	**Correct Predictions**	**Accuracy (%)**
Insufficient_Weight	317/346	91.6%
Normal_Weight	447/496	90.1%
Overweight_Level_I	353/376	93.9%
Overweight_Level_II	347/374	92.8%
Obesity_Type_I	396/440	90.0%
Obesity_Type_II	445/469	94.9%
Obesity_Type_III	603/606	**99.5%**
Overall	**2,828/3,107**	**91.02%**

Our key findings:

- **Best Performance:** Obesity_Type_III (99.5% accuracy) - most severe obesity class is highly distinguishable
- **Challenging Classes we have found:**
  - Obesity_Type_I (90.0%) - may likely be confused with adjacent categories
  - Normal_Weight (90.1%) - boundary cases with Insufficient_Weight and Overweight_Level_I
- **Confusion Pattern:** Misclassifications primarily occurs between the adjacent weight categories (e.g., Overweight_Level_I Overweight_Level_II), which reflects the continuous nature of weight progression
- **Consistent Performance:** All classes achieve ->90% accuracy, demonstrating a balanced model performance across the obesity spectrum

7.3 Performance comparison across approaches

Table 14: Complete Model Performance Comparison

Approach	**Validation Acc.**	**Test Acc.**	**Improvement**
Logistic Regression	0.8375	–	Baseline
Decision Tree	0.8404	–	+0.29%
Random Forest	0.8838	–	+4.63%
XGBoost (baseline)	0.8989	–	+6.14%
XGBoost (Optuna-tuned)	0.9115	**0.91487**	**+11.12%**

Performance Progression:

1. **Linear Model Baseline:** Logistic Regression (83.75%)
2. **Simple Tree Model:** Decision Tree (+0.29% improvement)
3. **Ensemble Method:** Random Forest (+4.63% improvement)
4. **Gradient Boosting:** XGBoost baseline (+6.14% improvement)
5. **Hyperparameter Optimization:** Optuna-tuned XGBoost (+11.12% total improvement)

7.4 Discussion of key factors

7.4.1 Feature engineering impact

One-Hot Encoding:

- We have transformed 8 categorical features into 22 binary indicators
- Then enabled XGBoost to learn non-linear interactions between the nominal categories
- Then avoided the ordinal assumptions for truly categorical variables (e.g., MTRANS, CALC)

Feature Importance Implications:

- Weight, Height, Age are expected to be top predictors (anthropometric measurements)
- Family history (FHWO) is likely a strong predictor (genetic/environmental factors)
- Lifestyle factors (FAF, TUE) contributes to the overall pattern recognition as well

7.4.2 Hyperparameter tuning impact

Regularization Strategy:

- The low gamma (0.384) prevents unnecessary splits
- Using moderate L2 regularization (lambda=0.033) smooths leaf weights
- Using low L1 regularization (alpha=0.990) retains most features

Tree Structure Optimization:

- max_depth=16 allows complex decision boundaries without overfitting
- min_child_weight=15.03 ensures sufficient samples per leaf
- n_estimators=606 provides adequate ensemble diversity

Sampling Strategy:

- subsample=0.609 uses most data per tree (high data efficiency)
- colsample_bytree=0.305 introduces feature diversity (this decorrelates trees)

7.4.3 Model generalization analysis

The 3.85 percentage point improvement from validation (91.15%) to test (91.487%) means that:

Positive Indicators:

- Strong regularization has prevented overfitting
- Stratified train-test split has maintained class balance
- Hyperparameter optimization has focused on generalization (not just training the accuracy)

Possible Explanations for Test Set Superiority:

- Test set may contain clearer class separations
- Validation set (20% of training) smaller than the actual Kaggle test set
- Model has effectively learned generalizable patterns rather than training-specific noise

8 Limitations and future work

8.1 Current limitations

- **Limited EDA:** No extensive exploratory data analysis performed on feature distributions, correlations, or outlier detection
- **No Feature Engineering:** Used raw features without creating derived features (e.g. BMI = Weight/Height^2, interaction terms)
- **Black-Box Model:** XGBoost provides limited interpretability compared to linear models
- **Computational Cost:** Optuna tuning with 50 trials requires significant computation time
- **Single Algorithm Focus:** Only XGBoost was tuned. other advanced methods (CatBoost, LightGBM, Neural Networks) not explored
- **Static Split:** Single train-test split (20%) rather than k-fold cross-validation for more robust performance estimates

8.2 Future work recommendations

1. **Ensemble methods:**
   - Voting/stacking ensemble of XGBoost, LightGBM, CatBoost
   - Weighted averaging based on individual model strengths
2. **Hyperparameter tuning enhancements:**
   - Increase Optuna trials (100-200)
   - Bayesian optimization with Gaussian Processes
   - Multi-objective optimization (accuracy + inference speed)