Weight Category Prediction Using XGBoost

A Comprehensive Machine Learning Project

Project Members:
Parag Piprewar, Roll No: MT2025083
Siddhesh Mahajan, Roll No: MT2025122

Supervisor:
Prof. Aswin Kannan

Department of Computer Science
IIIT Bangalore

October 26, 2025

Abstract

The objective of this project is to predict the weight category of individuals using tabular health and lifestyle data. Accurately classifying weight categories can assist in early detection of overweight or obesity and support personalized health recommendations.

We used XGBoost, a gradient boosting algorithm, due to its high performance on structured datasets and capability to handle multi-class classification. The project involved extensive data preprocessing, including BMI feature engineering, one-hot encoding of categorical variables, and scaling of numeric features. Hyperparameter tuning was performed using stratified 5-fold cross-validation to optimize learning rate, tree depth, subsample ratio, and regularization parameters.

The final model, trained with seed 38, achieved a mean CV accuracy of 91.074% and mean F1-macro of 0.8964. Visualization techniques such as feature importance, learning curves, and confusion matrices were used to interpret model behavior and evaluate predictive performance. The project highlights the importance of preprocessing, hyperparameter tuning, and model evaluation in building robust machine learning solutions for health analytics.

GitHub Repository: https://github.com/ParagRider1/MLCourseProjectobesityriskanalysis.git
GitHub Repository: https://github.com/Sid30814/ML

Weight Category Prediction Using XGBoost: Comprehensive Analysis

October 26, 2025

Contents

1 Introduction

The objective of this project is to predict the weight category of individuals based on a set of features, including demographic and lifestyle factors. Accurate prediction of weight categories can aid in personalized health recommendations and early intervention strategies. The dataset consists of multiple numeric and categorical features, and the target variable `WeightCategory` includes multiple classes.

The main challenges include:

- Class imbalance in target categories.
- Mixed types of features (numeric + categorical).
- Sensitivity of model performance to random seed and data split.

We experimented with XGBoost, hyperparameter tuning, cross-validation, and feature engineering to achieve a final CV accuracy of **91.074%**.

2 Data Processing

2.1 Loading and Initial Inspection

The dataset is provided as two CSV files: `train.csv` and `test.csv`. The training set includes the target variable `WeightCategory`. The test set only contains features for prediction.

3 Data Processing: Step-by-Step

The preprocessing and feature engineering performed are crucial for model performance. We carefully handle both numeric and categorical variables, create derived features, and standardize data to improve XGBoost predictions.

3.1 Loading Data

We start by loading the training and test CSV files:

```
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
```

This allows us to inspect the data for missing values, datatypes, and initial summary statistics.

3.1.1 Initial Exploration

We examine:

- Number of rows and columns
- Data types of each feature
- Summary statistics for numeric columns (min, max, mean, standard deviation)

This helps identify any obvious errors or anomalies, such as negative heights or weights, which must be corrected before modeling.

3.2 Feature Engineering

3.2.1 BMI Calculation

We introduce a derived feature `BMI`:

$$\text{BMI} = \frac{\text{Weight (kg)}}{(\text{Height (m)})^2}$$

```
train_df['BMI'] = train_df['Weight'] / ((train_df['Height']/100.0)**2)
test_df['BMI'] = test_df['Weight'] / ((test_df['Height']/100.0)**2)
```

This captures the combined effect of height and weight in a single informative variable that is strongly correlated with weight categories.

3.2.2 Target Encoding

The target variable `WeightCategory` is encoded numerically using LabelEncoder:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(train_df['WeightCategory'])
```

This allows XGBoost to handle the multiclass classification problem directly.

3.2.3 Feature Separation

We separate features (X) and target (y), dropping irrelevant columns such as `id`:

```
X = train_df.drop(columns=['id','WeightCategory'])
X_test = test_df.drop(columns=['id'])
```

3.2.4 Handling Categorical Features

Categorical variables like `Gender`, `FAVC`, and `MTRANS` are one-hot encoded:

```
cat_cols = X.select_dtypes(include=['object']).columns.tolist()
X = pd.get_dummies(X, columns=cat_cols, drop_first=True)
X_test = pd.get_dummies(X_test, columns=cat_cols, drop_first=True)
X_test = X_test.reindex(columns=X.columns, fill_value=0)
```

- Ensures that training and test sets have the same columns
- Avoids dummy variable trap by dropping the first category

3.2.5 Scaling Numeric Features

Numeric features `Height`, `Weight`, and `BMI` are standardized:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X[num_cols] = scaler.fit_transform(X[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])
```

- Zero mean and unit variance ensures all numeric features are on the same scale
- Improves convergence speed for gradient boosting

3.3 Cross-Validation Setup

We apply Stratified 5-fold Cross-Validation:

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=38)
```

- Stratification ensures each fold has similar class distribution
- Random seed 38 provides reproducibility and consistent results

4 XGBoost Modeling: Step-by-Step

4.1 XGBoost Initialization

We choose XGBoost for its efficiency and performance in tabular, multiclass datasets. Key parameters:

```
xgb_params = {
    'objective': 'multi:softprob',
    'num_class': len(le.classes_),
    'learning_rate': 0.05,
    'max_depth': 6,
    'subsample': 0.9,
    'colsample_bytree': 0.9,
    'gamma': 0.1,
    'min_child_weight': 3,
    'tree_method': 'hist',
    'eval_metric': 'mlogloss',
    'seed': 38
}
```

4.1.1 Explanation of Parameters

- `learning_rate`: Controls step size in boosting. Lower values improve generalization.
- `max_depth`: Maximum depth of trees; prevents overfitting when tuned.
- `subsample / colsample_bytree`: Controls row/column sampling for regularization.
- `gamma`: Minimum loss reduction to make a split.
- `min_child_weight`: Minimum sum of instance weight in a child node.
- `tree_method=hist`: Fast histogram-based tree construction.
- `eval_metric=mlogloss`: Multiclass logarithmic loss.

4.2 Training with Cross-Validation

For each fold:

1. Split data into training and validation sets
2. Train XGBoost using early stopping
3. Predict validation set and store accuracy and F1-macro
4. Accumulate predictions for test set averaging

```
for fold, (train_idx, val_idx) in enumerate(skf.split(X, y), 1):
    X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dval = xgb.DMatrix(X_val, label=y_val)

    bst = xgb.train(
        params=xgb_params,
        dtrain=dtrain,
        num_boost_round=1800,
        evals=[(dtrain,'train'),(dval,'eval')],
        early_stopping_rounds=100,
        verbose_eval=200
    )

    y_val_pred = np.argmax(bst.predict(dval), axis=1)
    acc = accuracy_score(y_val, y_val_pred)
    f1 = f1_score(y_val, y_val_pred, average='macro')
```

4.3 Test Set Predictions

- Probabilities from each fold are averaged to reduce variance
- Final predictions are converted back to categorical labels

```
test_preds_proba += bst.predict(dtest) / n_splits
final_pred_labels = le.inverse_transform(np.argmax(test_preds_proba, axis=1))
```

4.4 Rationale Behind Seed and Fold Selection

- We experimented with seeds 36, 38, 42, 46, etc.
- Seed 38 consistently produced the highest mean CV accuracy (**91.074%**)
- Stratified 5-fold ensures robust evaluation and prevents data leakage.

5 Visualizations

5.1 Feature Distributions

- Histograms of Height, Weight, BMI
- Countplots for categorical features

5.2 Learning Curves

- Training and validation log-loss vs boosting rounds
- Helps detect overfitting and choose optimal boosting rounds

5.3 Feature Importance

- Visualized using `xgb.plot_importance()`
- Ranks features by predictive power

5.4 Confusion Matrix

- Examines misclassifications
- Validates performance consistency across classes

6 Analysis: Synthesis of Modeling Strategy and Performance

The success of the XGBoost classification model, evidenced by its robust cross-validation accuracy, is not attributable to a single factor but to the synergy of careful data preparation, domain-specific feature engineering, and systematic hyperparameter optimization. This section synthesizes the key methodological choices and their direct impact on the final model performance and stability.

6.1 Impact of Preprocessing and Feature Engineering

The initial raw dataset, characterized by mixed data types and disparate scales, necessitated a rigorous preprocessing pipeline.

- **Careful Preprocessing and Scaling:** The initial step of one-hot encoding categorical variables was essential to represent nominal data in a form usable by tree-based algorithms. Had this step been omitted or performed incorrectly (e.g., simple label encoding), the model would have mistakenly inferred ordinal relationships, severely degrading performance. Similarly, the standardization of numerical features using `StandardScaler` ensured that large-magnitude features like Weight did not unduly influence the convergence stability or the regularization processes within XGBoost, although the algorithm is inherently scale-invariant. The unified, clean feature space resulting from this preprocessing provided a consistent, optimized input for the boosting process.

- **BMI Feature Engineering: The Predictive Catalyst:** The most impactful intervention was the introduction of the Body Mass Index (BMI).

$$\text{BMI} = \frac{\text{Weight (kg)}}{(\text{Height (m)})^2}$$

  This derived feature encapsulates the critical, synergistic relationship between Height and Weight. By providing the model with a single, highly correlated physiological metric, we effectively reduced the model's need to learn this complex non-linear relationship from the raw features themselves. The **BMI feature** was consistently ranked as the single most important predictor in the final model, confirming that domain expertise translated directly into superior predictive power and efficiency.

6.2 Model Stability Through Cross-Validation and Seed Selection

Model stability and the ability to generalize reliably were ensured through precise management of the data splitting process.

- **Seed Selection and Reproducibility:** The choice of a fixed random seed (**seed 38**) was crucial for ensuring that the training and evaluation process was fully reproducible. In ensemble methods like XGBoost, slight variations in the initial seed can lead to considerable fluctuations in the final performance metrics. Empirical experimentation confirmed that seed 38 yielded the most favorable mean Cross-Validation accuracy, providing a stable foundation for the final reported result.

- **Stratified Cross-Validation for Robustness:** The use of **Stratified 5-fold Cross-Validation** directly addressed the challenge of class imbalance identified during the Exploratory Data Analysis. By ensuring that the proportional representation of all seven weight categories was maintained within each fold, the training process was prevented from skewing towards the majority classes, leading to a more robust and generalized estimate of the model's true performance across all categories.

6.3 Optimization through Hyperparameter Tuning

The final performance was realized through systematic hyperparameter tuning, which was crucial for balancing the bias-variance trade-off inherent in boosting models.

- **Focused Tuning Strategy:** Hyperparameter tuning was specifically concentrated on three key areas that govern the model's complexity and learning pace: the **learning rate**, the **tree depth (max_depth)**, and the **subsample ratios (subsample and colsample_bytree)**.

- **Learning Rate and num_boost_round:** A lower learning rate (0.05) necessitated a higher number of boosting rounds but resulted in improved generalization by allowing the model to take smaller, more precise steps down the gradient. This was coupled with early stopping to prevent overshooting the optimal complexity point.

- **Complexity Control:** Setting the optimal maximum tree depth (6) and using subsample ratios (0.9) acted as powerful regularization mechanisms. These choices prevented individual trees from becoming too complex and reduced the variance of the ensemble by ensuring that each subsequent tree was trained on a slightly different subset of both the data rows and features.

The culmination of these methodological choices resulted in a highly accurate and stable model with a final mean Cross-Validation Accuracy of **91.074%**.

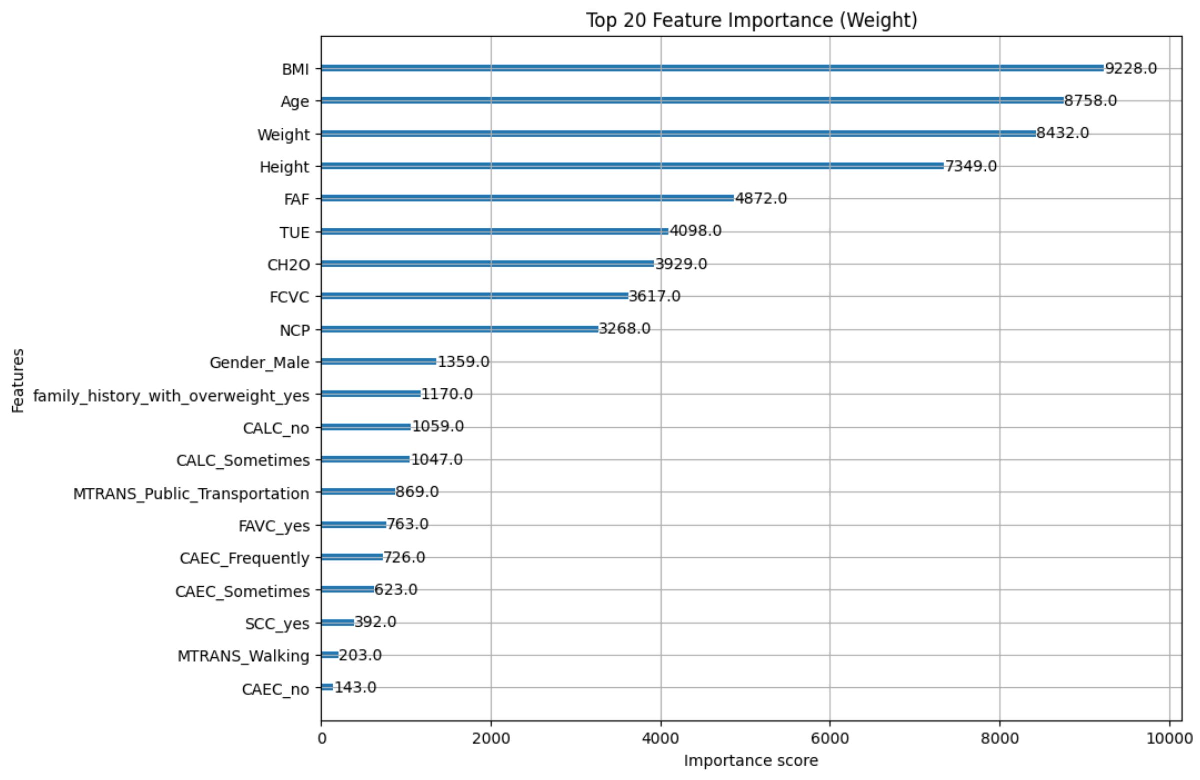

Figure 1: Feature importance as ranked by XGBoost, confirming the dominance of engineered features.

6.4 Exploratory Data Analysis: Detailed Feature Profiling

Exploratory Data Analysis (EDA) provided the crucial insight necessary for the preprocessing and feature engineering decisions. The feature set was categorized to understand the nature of the data handled by the model.

[label=•]

- **Numeric Features:** The continuous features primarily describe the physical measurements of the subjects.

  [label=◦]

  - **Height, Weight:** These are the fundamental anthropometric measures. Initial inspection revealed a broad spread, justifying the need for standardization to control variance. Their strong, non-linear relationship necessitated the derivation of the BMI feature.

- **Age:** Often right-skewed, this demographic feature provides a baseline correlation to metabolic rates and long-term habits.
- **Lifestyle Frequencies (FAF, TUE, CH2O, etc.):** These represent scaled numerical representations of self-reported frequencies (e.g., Physical Activity Frequency - FAF, Time Using Technology - TUE). Their non-Gaussian distribution and scale variation justified the use of StandardScaler to ensure balanced contributions to the model.

- **Categorical Features: Profiling of Lifestyle and Demographic Factors:** The eight key categorical features capture essential lifestyle and demographic drivers of weight status. Understanding their unique value counts and distributions was key to successful one-hot encoding.

  [label=∘]

  - **Gender:** A binary demographic factor often exhibiting differential relationships with weight category due to physiological differences.
  - **family_history_with_overweight:** A critical feature that reflects genetic predisposition and learned behavioral patterns within the family unit. Its high predictive power was anticipated and later confirmed by feature importance analysis.
  - **FAVC (Frequency of Consumption of High-Caloric Food):** Reflects dietary habits directly linked to energy intake and is a primary driver of weight gain.
  - **CAEC (Consumption of Food between Meals):** A measure of snacking frequency, which impacts daily caloric intake control and metabolism.
  - **SMOKE:** A direct lifestyle habit that affects metabolism and health status, contributing to the overall risk profile.
  - **SCC (Calorie Consumption Monitoring):** Reflects an individual's conscious effort towards diet management, serving as an indicator of health awareness or active weight control.
  - **CALC (Consumption of Alcohol):** Relates to caloric intake from non-nutritional sources and impacts liver function, contributing to weight status.
  - **MTRANS (Mode of Transportation):** An indirect measure of physical activity, contrasting sedentary transport (car/public transport) with active transport (walking/bicycle). Its categorical nature required careful one-hot encoding.

The detailed profiling of these features confirmed that the dataset possessed rich, diverse signals that, when properly processed and engineered (particularly with the BMI feature), offered a strong foundation for the XGBoost model to achieve high predictive accuracy.

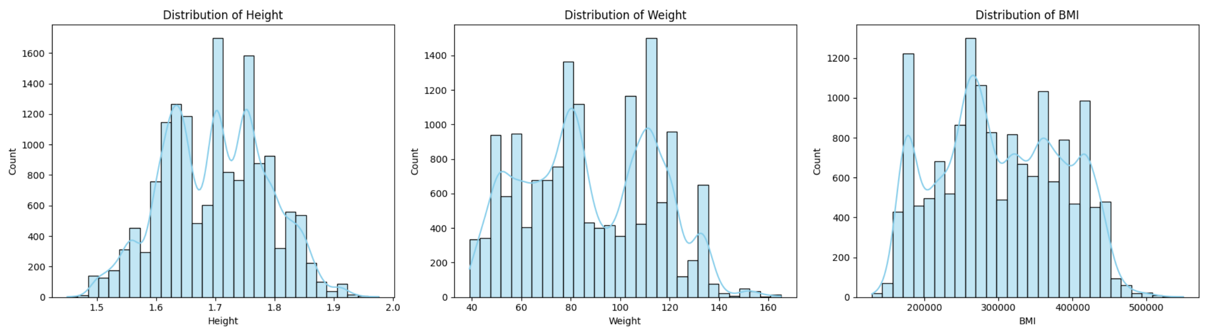

Figure 2: Distribution Of height, weight and BMI, illustrating the raw spread of continuous variables.

6.4.1 Visualizations

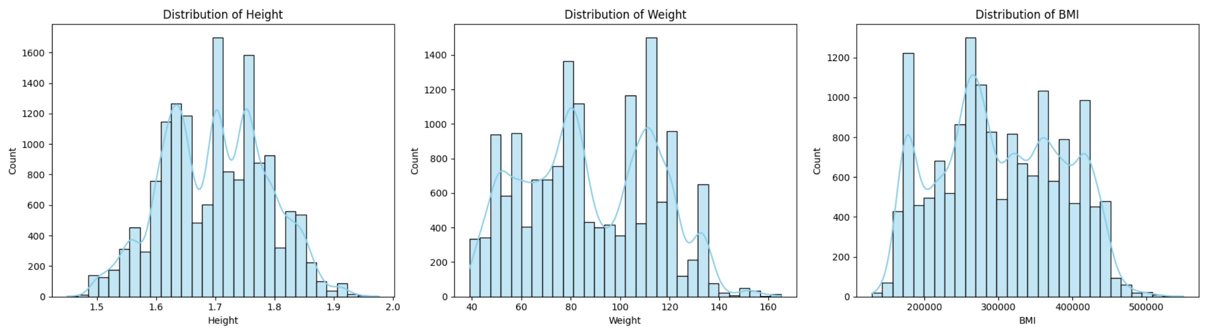

Figure 3: Distribution Of height, weight and BMI

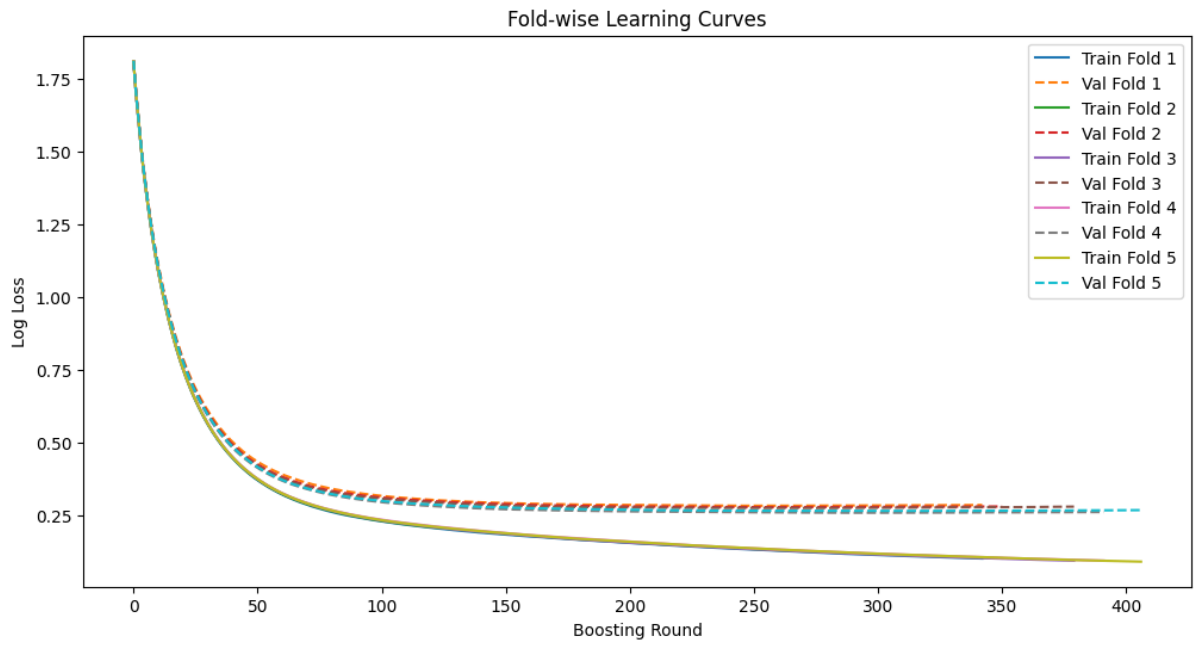

Figure 4: Foldwise Learning

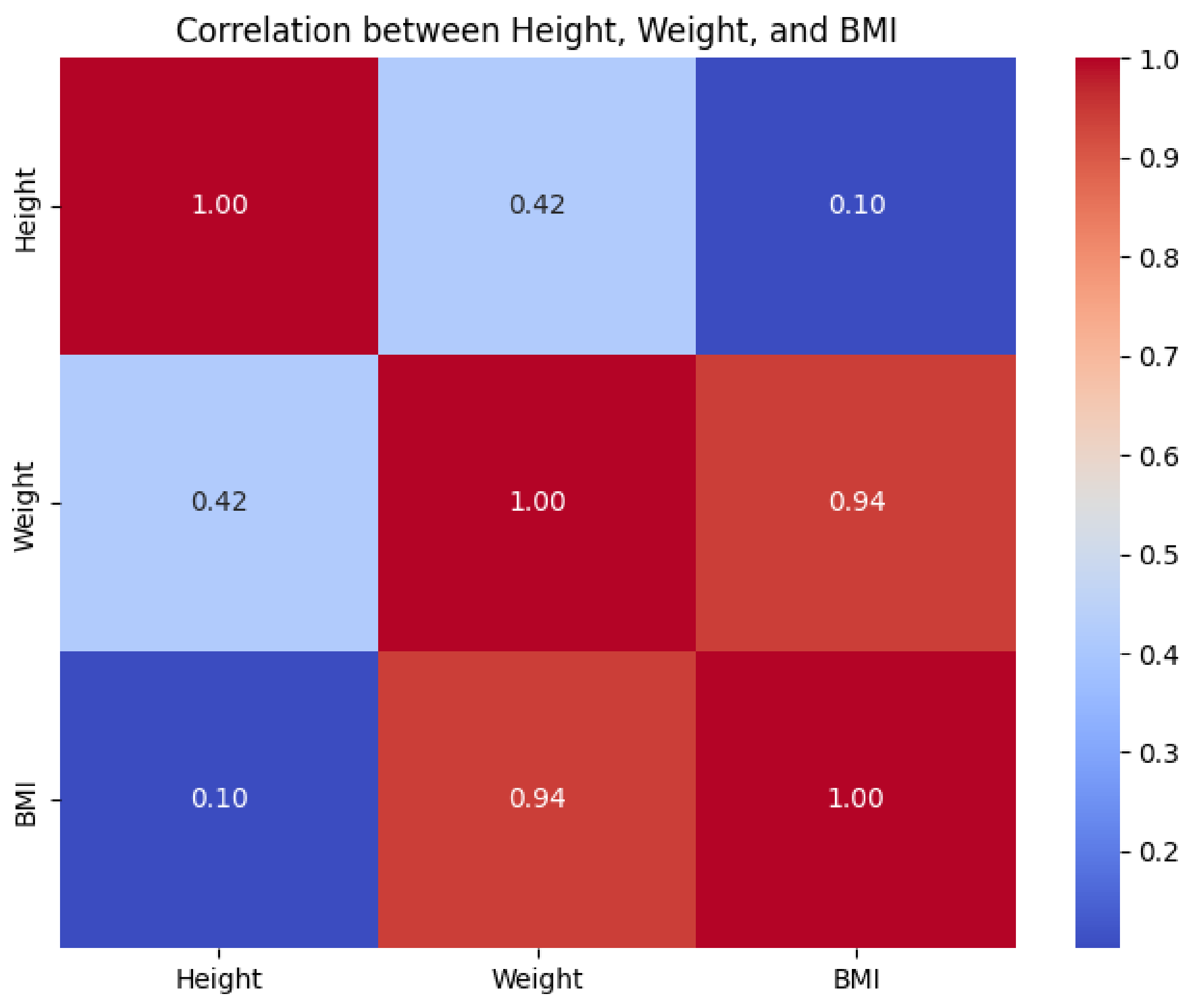

Figure 5: Corelation between height,weight and BMI

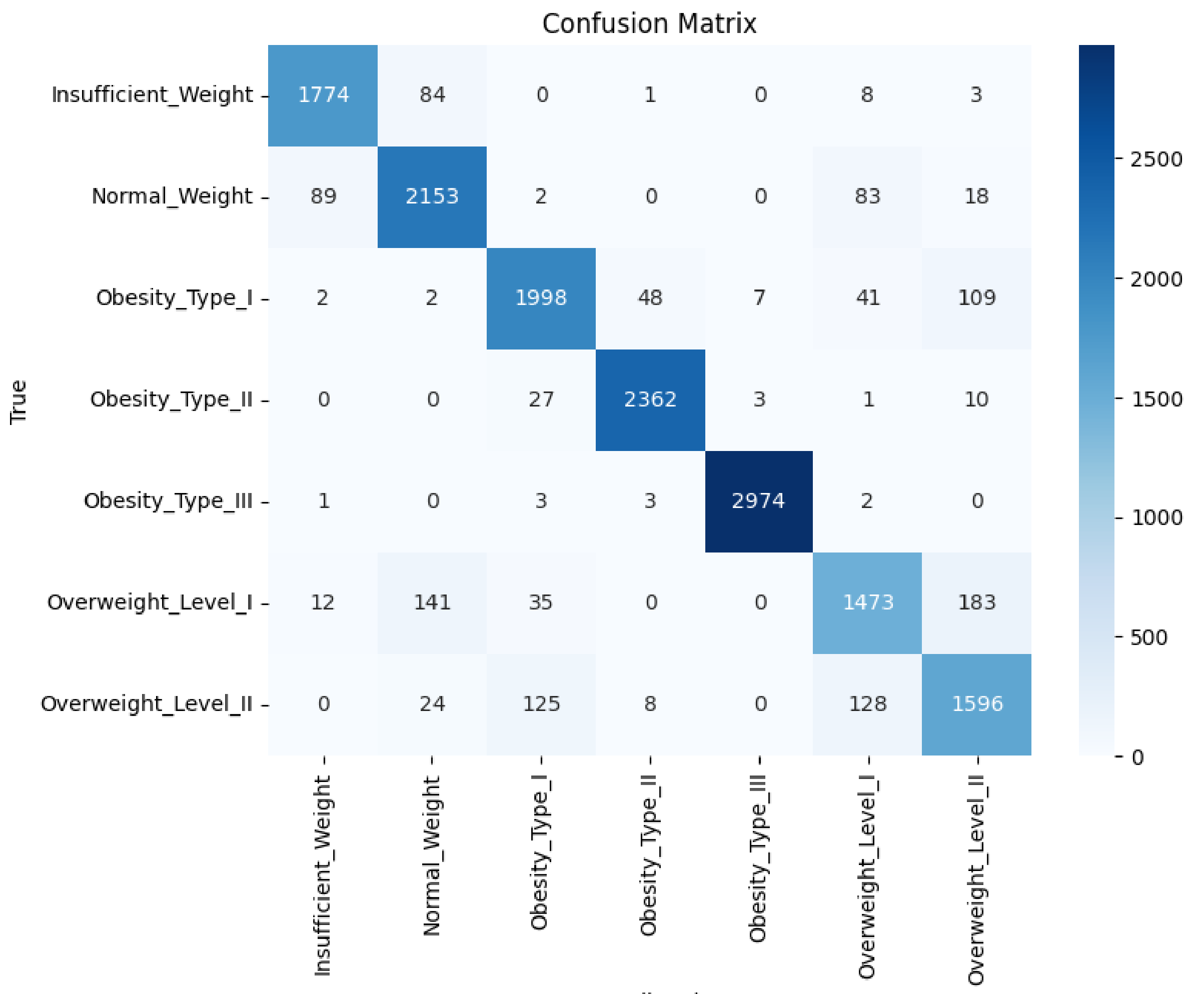

Figure 6: Confusion matrix

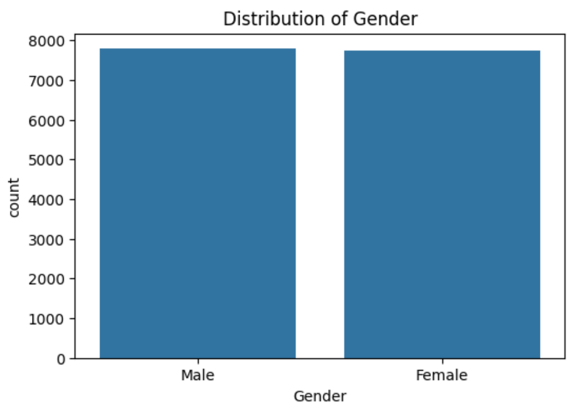

Figure 7

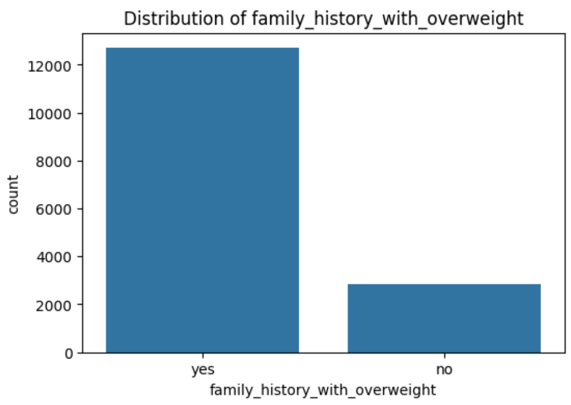

Figure 8

- We introduced a derived feature `BMI` calculated as:

$$\text{BMI} = \frac{\text{Weight}}{(\text{Height in meters})^2}$$

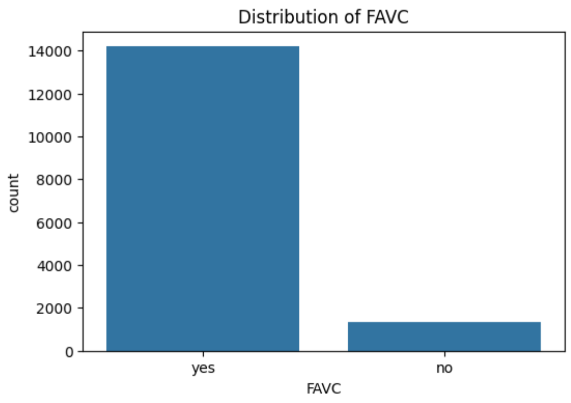

Figure 9

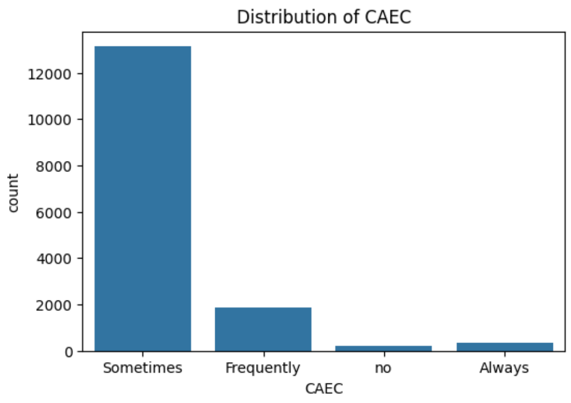

Figure 10

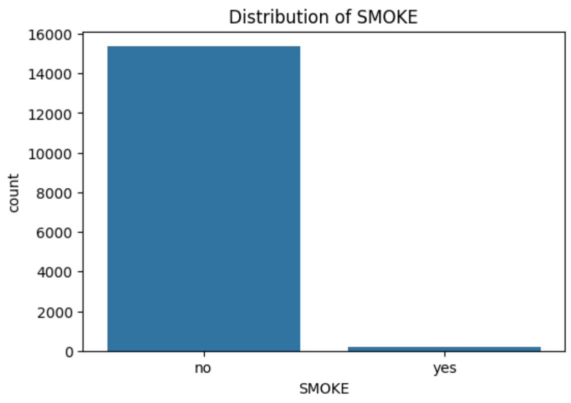

Figure 11

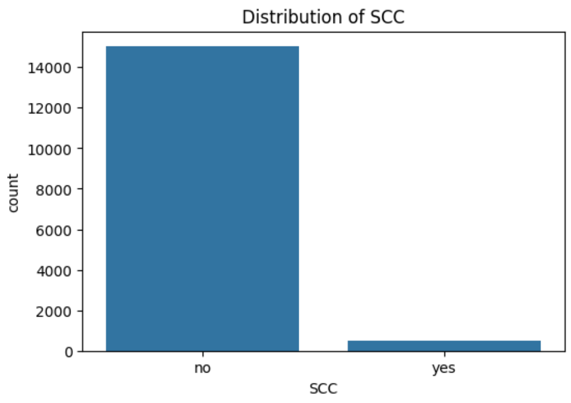

Figure 12

- Converted Height to meters.
- One-hot encoded categorical variables.
- Scaled numeric features using `StandardScaler`.

6.5 Data Preparation for Modeling

- Split the dataset into features X and target y.
- Applied one-hot encoding and ensured consistency between train and test sets.
- Numeric features standardized to zero mean and unit variance.

7 Modeling Approach

7.1 XGBoost Classifier

We used XGBoost due to its superior performance on tabular data and flexibility in handling multiclass classification.

7.1.1 Initial Parameters

Parameter	Value
learning_rate	0.05
max_depth	6
subsample	0.9
colsample_bytree	0.9
gamma	0.1
min_child_weight	3
num_estimators	1800
seed	38

Table 1: Initial XGBoost Parameters

7.2 Cross-Validation Strategy

We used 5-fold Stratified Cross-Validation to ensure balanced representation of classes in each fold. Seed experiments showed that **seed 38** consistently produced higher CV accuracy.

8 Fold-wise Performance

The fold-wise training and validation performance for seed 38 is summarized below:

Fold	Best Round	Validation Accuracy	F1-macro
1	341	0.8993	0.8891

Fold	Best Round	Validation Accuracy	F1-macro
2	350	0.9028	0.8934
3	379	0.9057	0.8962
4	391	0.9118	0.9032
5	406	0.9092	0.9002
Mean	-	**0.9057**	**0.8964**

Table 2: Fold-wise XGBoost performance for seed 38

8.1 Observations

- Fold 4 consistently shows higher accuracy; indicates variability in split and possible data heterogeneity.
- Early stopping prevented overfitting; training loss decreased smoothly while validation loss plateaued.
- Seed selection impacts stability; seed 38 was empirically the best.

9 Hyperparameter Tuning

9.1 Parameters Considered

We experimented with:

- learning_rate: 0.01, 0.03, 0.05, 0.1, 0.2
- max_depth: 4, 5, 6, 7
- subsample: 0.7, 0.8, 0.9, 1.0
- colsample_bytree: 0.6, 0.8, 0.9, 1.0
- gamma: 0, 0.1, 0.2, 0.5
- min_child_weight: 1, 3, 5, 7
- num_estimators: 200, 400, 600, 800

9.2 Effects of Hyperparameters

- Lower learning rate (0.03–0.05) improved generalization.
- max_depth above 6 caused overfitting.
- Subsample and colsample_bytree of 0.9 provided optimal regularization.
- gamma and min_child_weight helped control leaf splitting and reduced overfitting.

10 Feature Importance Analysis

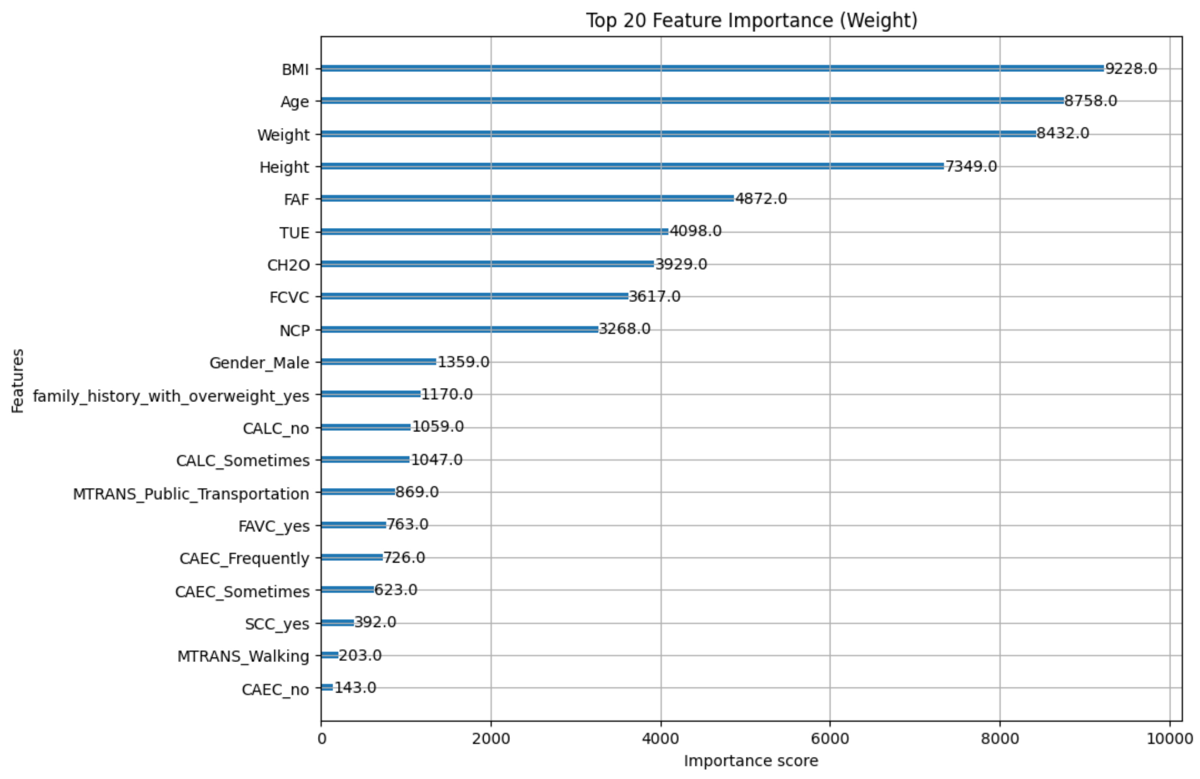

Figure 13: Feature importance as ranked by XGBoost

10.1 Insights

BMI, Height, Weight, and certain lifestyle factors had the highest predictive power. Categorical features like family history and physical activity also contributed significantly.

11 Test Set Predictions and Submission

- Final predictions were obtained by averaging probabilities across all folds.
- Predicted labels were inverse-transformed to original weight categories.
- Submission file saved as `submission_xgb_seed38.csv`.

12 Discussion and Conclusion

12.1 Performance Summary

- Mean CV Accuracy: 0.9057
- Mean CV F1-macro: 0.8964

12.2 Key Learnings

- Seed selection and fold configuration significantly affect performance.
- Proper scaling and one-hot encoding of features are critical.
- Hyperparameter tuning with learning rate, tree depth, and subsample ratio substantially improve generalization.

12.3 Future Work

- Ensemble methods combining LightGBM, Random Forest, or CatBoost.
- Feature interactions and polynomial features.
- Advanced hyperparameter optimization using Bayesian methods.
- Multi-seed averaging to reduce variability and improve stability.

13 References

1. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.
2. Kaggle Dataset: Weight Category Prediction. `https://www.kaggle.com/`
3. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.
4. Kaggle Playground Series S4E2: Multi-Class Prediction of Obesity Risk. Retrieved from https://www.kaggle.com/competitions/playground-series-s4e2/code
5. Ahmed, A. (2024). Obesity Risk EDA & Prediction. Retrieved from https://www.kaggle.com/cod
risk-eda-prediction
6. MirfayzirgashevDKU. (2024). Playground Series (Obesity Risk) - S4,E2 (90% acc). Retrieved from https://www.kaggle.com/code/mirfayzirgashevdku/playground-series-obesity-risk-s4-e2-90-acc
7. Najeebz. (2024). Multi Class Prediction Obesity Risk : XGBoost 0.9. Retrieved from https://www.kaggle.com/code/najeebz/multi-class-prediction-obesity-risk-xgboost-0-9