Obesity Risk Prediction(Kaggle)

Course: AIT511 - Machine Learning

Team Members: MT2025008 MT2025059
Department of Computer Science and Engineering
IIIT-Bangalore

Date: October 28, 2025

Algorithms Used:
Multiple Regression, AdaBoost, XGBoost

Contents

1 Overview

Maintaining a healthy lifestyle has become increasingly challenging in modern society. This project focuses on analyzing how an individual's weight category is influenced by their daily habits, food choices, physical activity, and demographic background.

The main goal is to build machine learning models that can correctly classify people into categories such as:

- Insufficient Weight
- Normal Weight
- Overweight
- Obesity Type I
- Obesity Type II
- Obesity Type III

The dataset includes information like age, gender, family history, dietary behavior, activity level, screen time, and transportation mode. Because these factors are diverse and interrelated, the task combines elements of machine learning, behavioral science, and healthcare.

We experimented with three models — Multiple Regression, AdaBoost, and XGBoost — to determine which performed best for obesity prediction.

- **Multiple Regression** helped explore the linear relationships between lifestyle variables and obesity risk. However, its accuracy was limited due to the complex, non-linear nature of human health data.
- **AdaBoost** improved performance by combining several weak classifiers into one strong model but was still sensitive to noisy data and imbalanced samples.
- **XGBoost**, a gradient boosting technique, outperformed both, offering high accuracy, strong regularization, and robust handling of complex data.

2 Data Preprocessing

2.1 Loading and Understanding the Dataset

The dataset was sourced from the AIT-511 Obesity Risk Kaggle Project, containing both training and test CSV files. Each record represents a person's demographic and lifestyle attributes, along with their obesity category.

2.2 Feature Engineering

To enhance the model's understanding of the relationship between height, weight, and obesity, several new features were derived:

- Body Mass Index (BMI) = Weight / (Height^2)
- Height^2 = Height $\times$ Height
- Weight/Height = Weight / Height

These engineered features made it easier for the model to capture proportional and non-linear relationships.

```
df_combined['BMI'] = df_combined['Weight'] / (df_combined['Height'] ** 2)
df_combined['Height^2'] = df_combined['Height'] ** 2
df_combined['Weight/Height'] = df_combined['Weight'] / df_combined['Height']
```

2.3 Handling Categorical Variables

Categorical features such as `Gender`, `family_history_with_overweight`, `FAVC`, and `MTRANS` were transformed using **one-hot encoding**, ensuring the algorithm could interpret them correctly without implying a false order.

```
df_combined = pd.get_dummies(df_combined, columns=categorical_cols,
drop_first=True)
```

This transformation expanded each category into binary columns, allowing the model to detect complex interactions effectively.

2.4 Feature Scaling using StandardScaler

Since algorithms like XGBoost perform better when features share a similar range, **StandardScaler** was applied to normalize all numeric columns. This ensured that every feature contributed equally during model training.

```
scaler = StandardScaler()
df_combined_scaled = scaler.fit_transform(df_combined)
```

2.5 Splitting Train–Test Data

After preprocessing, the data was divided into training and test subsets. The target column (`WeightCategory`) was **label-encoded** into numerical classes (0–5) to make it compatible with machine learning models.

2.6 Final Preprocessed Data Summary

- All categorical features converted to numeric.
- New BMI-based features added.
- Data normalized using standard scaling.
- Final dataset ready for training and hyperparameter optimization.

3 Exploratory Data Analysis (EDA)

EDA helped uncover trends and patterns that could affect obesity levels.

3.1 Target Variable Distribution

A bar plot was used to visualize the number of samples in each weight category. This revealed that while most individuals fall into "Normal" or "Overweight" categories, a smaller proportion belongs to "Obesity Type II/III".

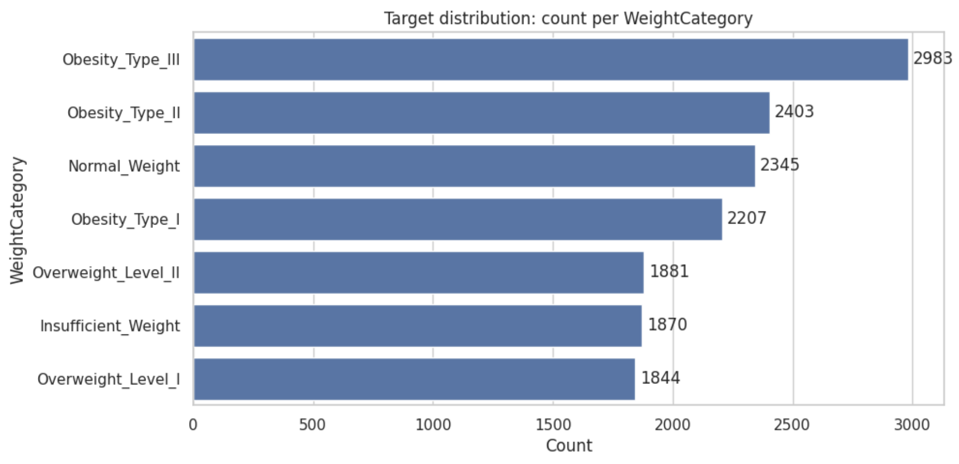


Figure 1: Target distribution: count per WeightCategory

This bar chart illustrates how the dataset is distributed across various weight categories, such as Normal Weight, Overweight, and Obesity Types I–III. It helps identify **class imbalance**, which is crucial for selecting appropriate evaluation metrics and stratified sampling techniques.

Target class percentages:

```
WeightCategory
Obesity_Type_III    19.20
Obesity_Type_II     15.47
Normal_Weight       15.10
Obesity_Type_I      14.21
Overweight_Level_II 12.11
Insufficient_Weight 12.04
Overweight_Level_I  11.87
```

Code snippet for plotting:

```
sns.countplot(x='WeightCategory', data=train_df)
plt.title('Distribution of Weight Categories')
plt.show()
```

Table 1: Descriptive Statistics of Numeric Features

Feature	**count**	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
id	15533.0	7766.000000	4484.135201	0.00	3883.000000	7766.000000	11649.000000	15532.0000
Age	15533.0	23.816308	5.663167	14.00	20.000000	22.771612	26.000000	61.000000
Height	15533.0	1.699918	0.087670	1.45	1.630927	1.700000	1.762921	1.975663
Weight	15533.0	87.785225	26.369144	39.00	66.000000	84.000000	111.600553	165.057269
FCVC	15533.0	2.442917	0.530895	1.00	2.000000	2.342220	3.000000	3.000000
NCP	15533.0	2.760425	0.706463	1.00	3.000000	3.000000	3.000000	4.000000
CH2O	15533.0	2.027626	0.607733	1.00	1.796257	2.000000	2.531456	3.000000
FAF	15533.0	0.976968	0.836841	0.00	0.007050	1.000000	1.582675	3.000000
TUE	15533.0	0.613813	0.602223	0.00	0.000000	0.566353	1.000000	2.000000

The boxplots compare how key numeric variables (like BMI and Age) vary across different obesity categories. These visualizations reveal patterns and differences between groups, assisting in feature relevance assessment.

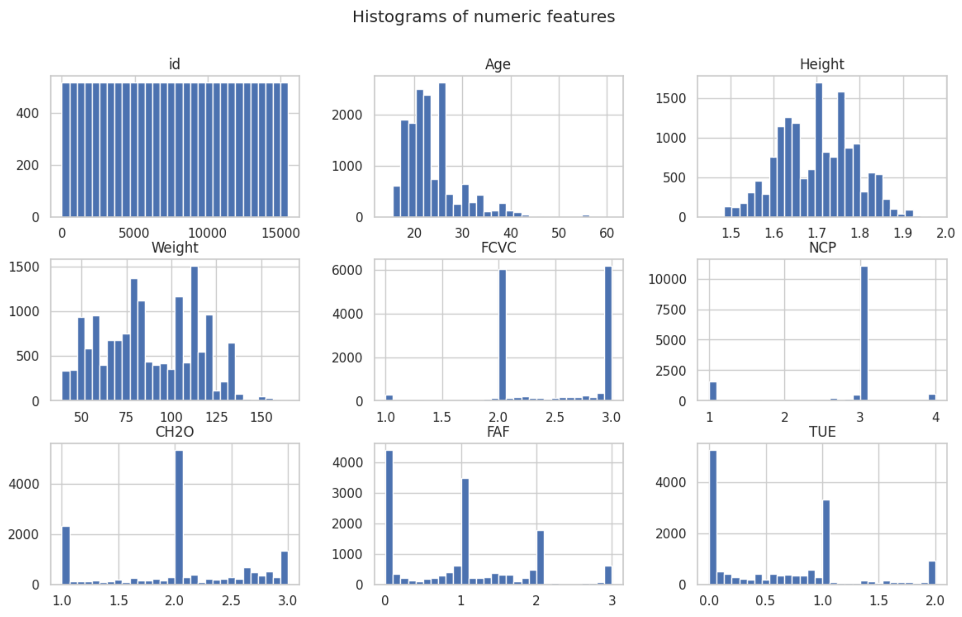


Figure 2: Numeric summary (describe) get central tendency, spread, and detect possible outliers

This plot visualizes the frequency of different categorical variables such as Gender, Food Consumption (FAVC), and Transport Mode (MTRANS). It gives a clearer understanding of how lifestyle factors are distributed across individuals.

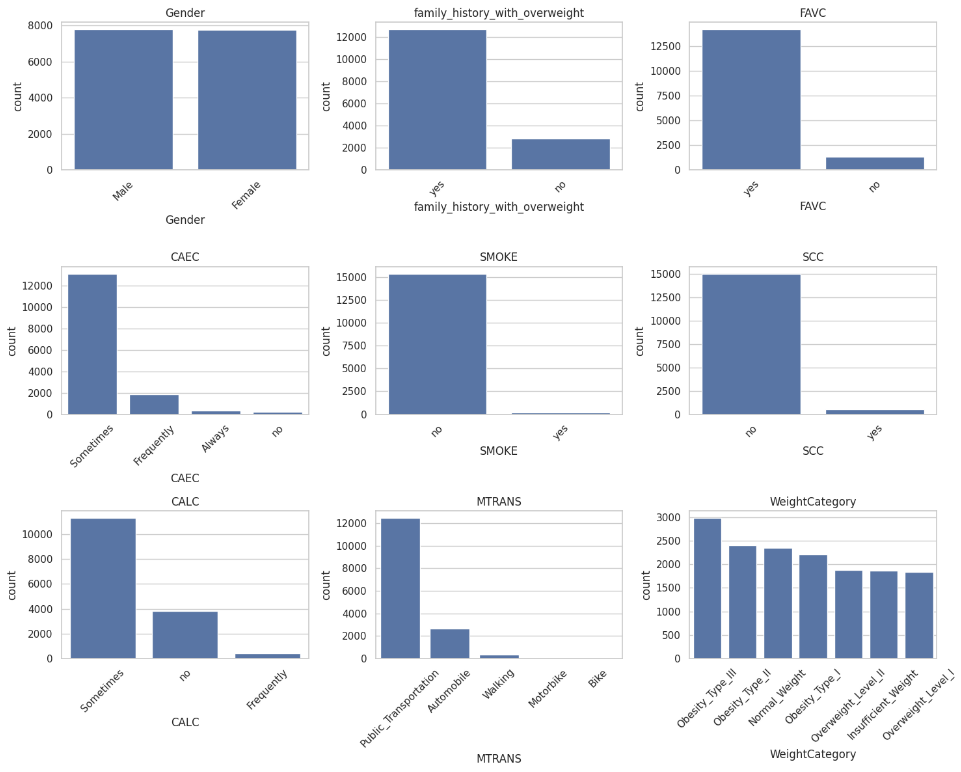


Figure 3: Categorical features distribution check distribution for features like Gender, FAVC, CAEC, MTRANS, etc.

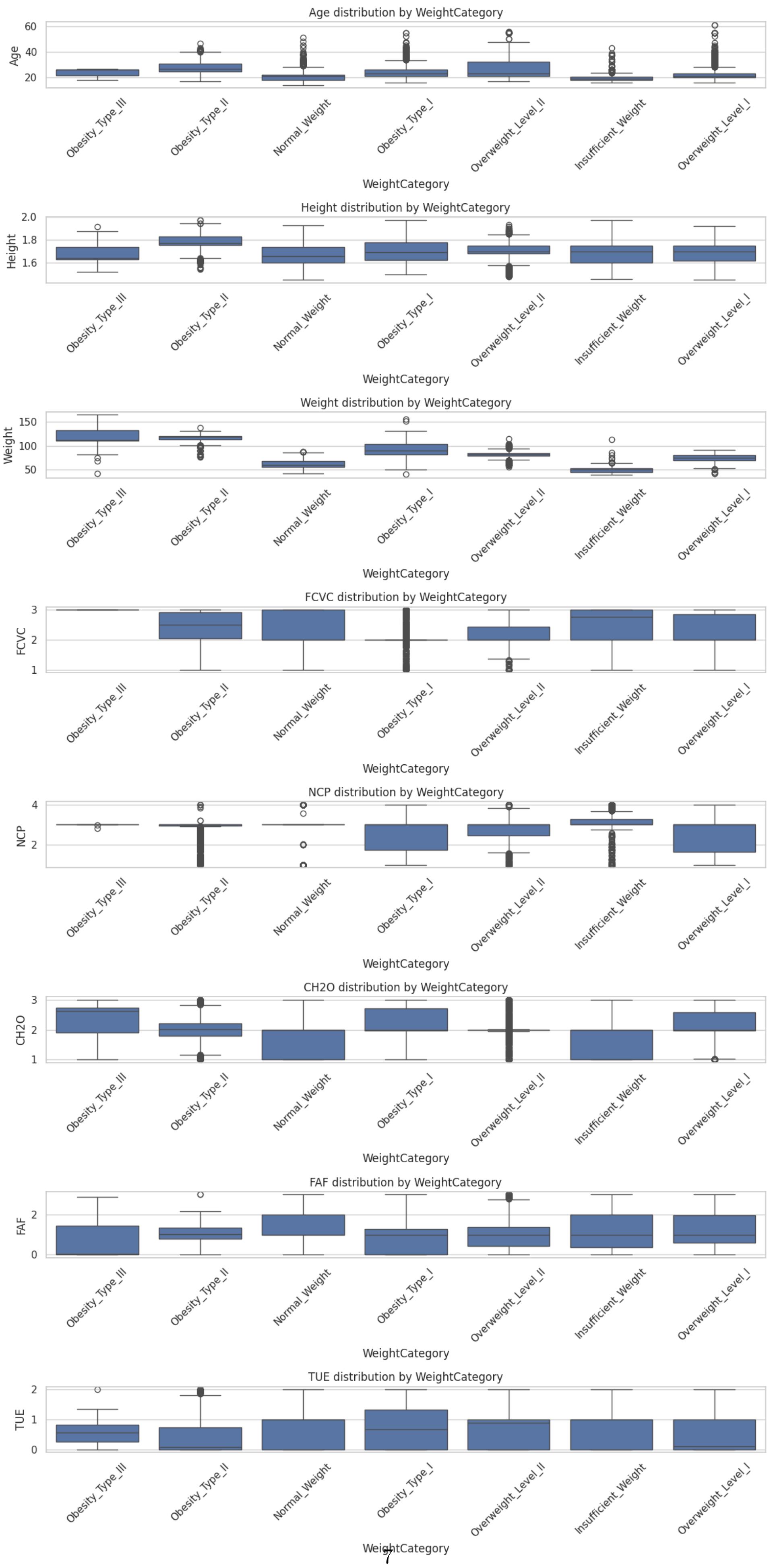


Figure 4: Relationship: numeric features vs target check how numeric features vary across categories (boxplots show distributions)

This pair plot shows relationships among selected numerical features. It helps visualize clustering tendencies and correlations that might influence weight categories.

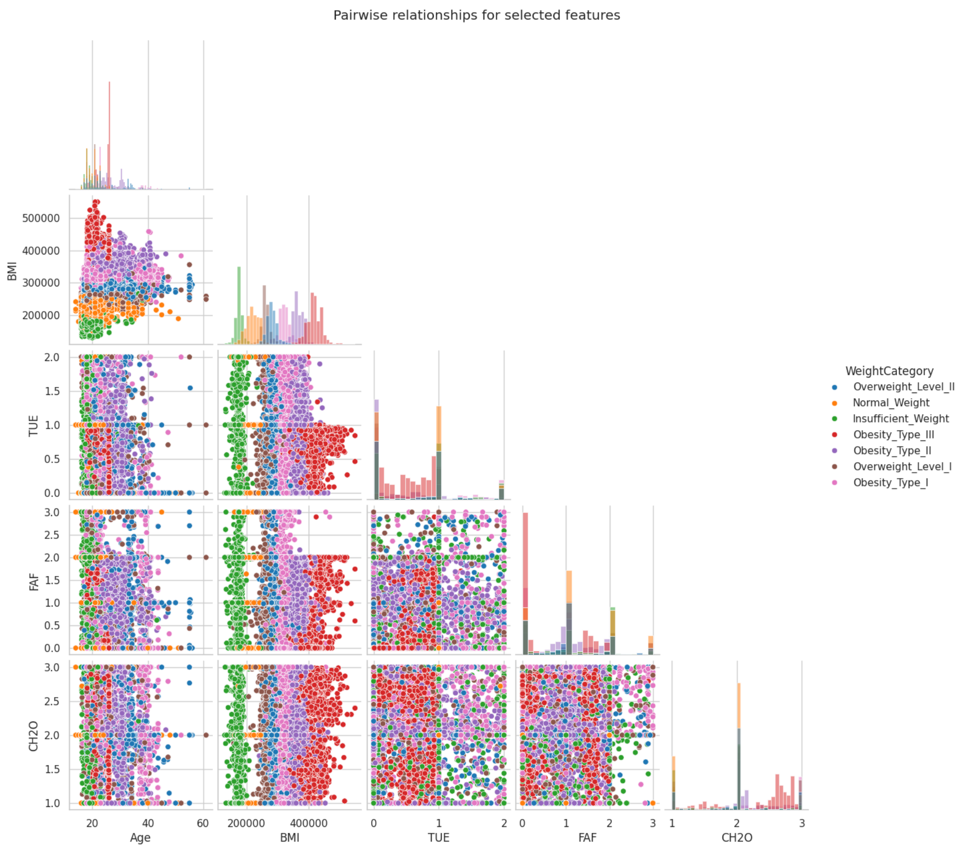
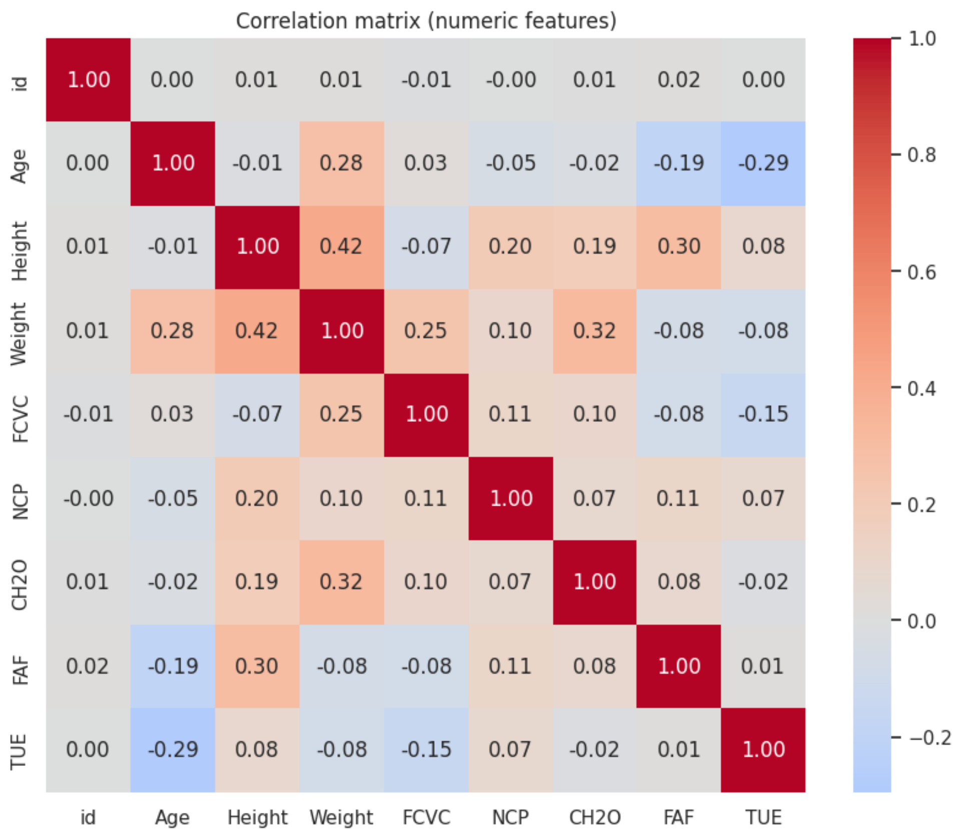


Figure 5: Pairwise relationships (small subset) pair plots are expensive with many features, pick a few important numeric ones.

The heatmap displays correlation coefficients among numeric variables such as Height, Weight, and BMI. A strong positive correlation between Weight and BMI confirms that BMI is a key indicator of obesity level.



	id	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
id	1.00	0.00	0.01	0.01	-0.01	-0.00	0.01	0.02	0.00
Age	0.00	1.00	-0.01	0.28	0.03	-0.05	-0.02	-0.19	-0.29
Height	0.01	-0.01	1.00	0.42	-0.07	0.20	0.19	0.30	0.08
Weight	0.01	0.28	0.42	1.00	0.25	0.10	0.32	-0.08	-0.08
FCVC	-0.01	0.03	-0.07	0.25	1.00	0.11	0.10	-0.08	-0.15
NCP	-0.00	-0.05	0.20	0.10	0.11	1.00	0.07	0.11	0.07
CH2O	0.01	-0.02	0.19	0.32	0.10	0.07	1.00	0.08	-0.02
FAF	0.02	-0.19	0.30	-0.08	-0.08	0.11	0.08	1.00	0.01
TUE	0.00	-0.29	0.08	-0.08	-0.15	0.07	-0.02	0.01	1.00

Figure 6: Correlation matrix (numeric) finds linear relationships between numeric variables; helps with feature selection/collinearity

3.2 Correlation Heatmap

To analyze relationships between numerical features (e.g., Weight, Height, BMI, FAF), a heatmap was generated.

Key insights:

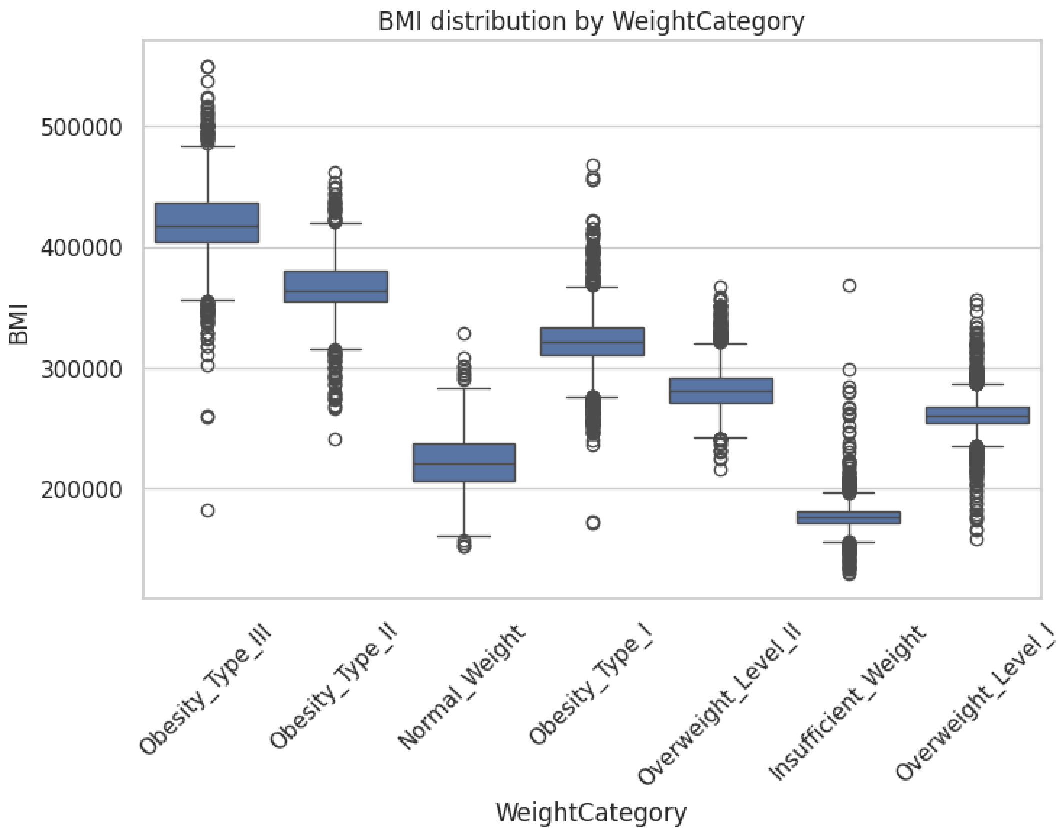
- Strong positive correlation between **BMI and Weight**.
- Negative correlation between **Height and Obesity level**.

These confirm that BMI and weight are key predictors.
BMI SUMMARY CLASS:

Table 2: BMI Summary by Weight Category

Weight Category	**Insufficient_Weight**	**Normal Weight**	**Obesity _Type_I**	**Obesity _Type_II**	**Obesity _Type_III**	**Overweight_Level_I**	**O t**
count	1870.000000	2345.000000	2207.000000	2403.000000	2983.000000	1844.000000	18
mean	176208.069429	220227.886205	321409.896045	365718.139940	418170.010105	260918.382493	282
std	14360.967040	21992.403195	25631.576473	21144.410780	26903.022934	18002.078553	17
min	128685.407075	150947.953146	170992.784105	240484.601993	181786.703601	157618.802836	215
25%	170373.404091	205693.296602	310204.081633	354453.927155	404162.831736	253906.250000	270
50%	175324.670676	220385.674931	320799.694079	363906.286912	417700.837480	259819.784973	280
75%	180620.379165	237118.446310	333205.177496	380716.248163	436348.210102	266727.632983	290
max	367781.150966	328824.141519	468051.876636	462224.831931	549979.913613	355555.555556	367

Figure 7: Correlation matrix (numeric features)

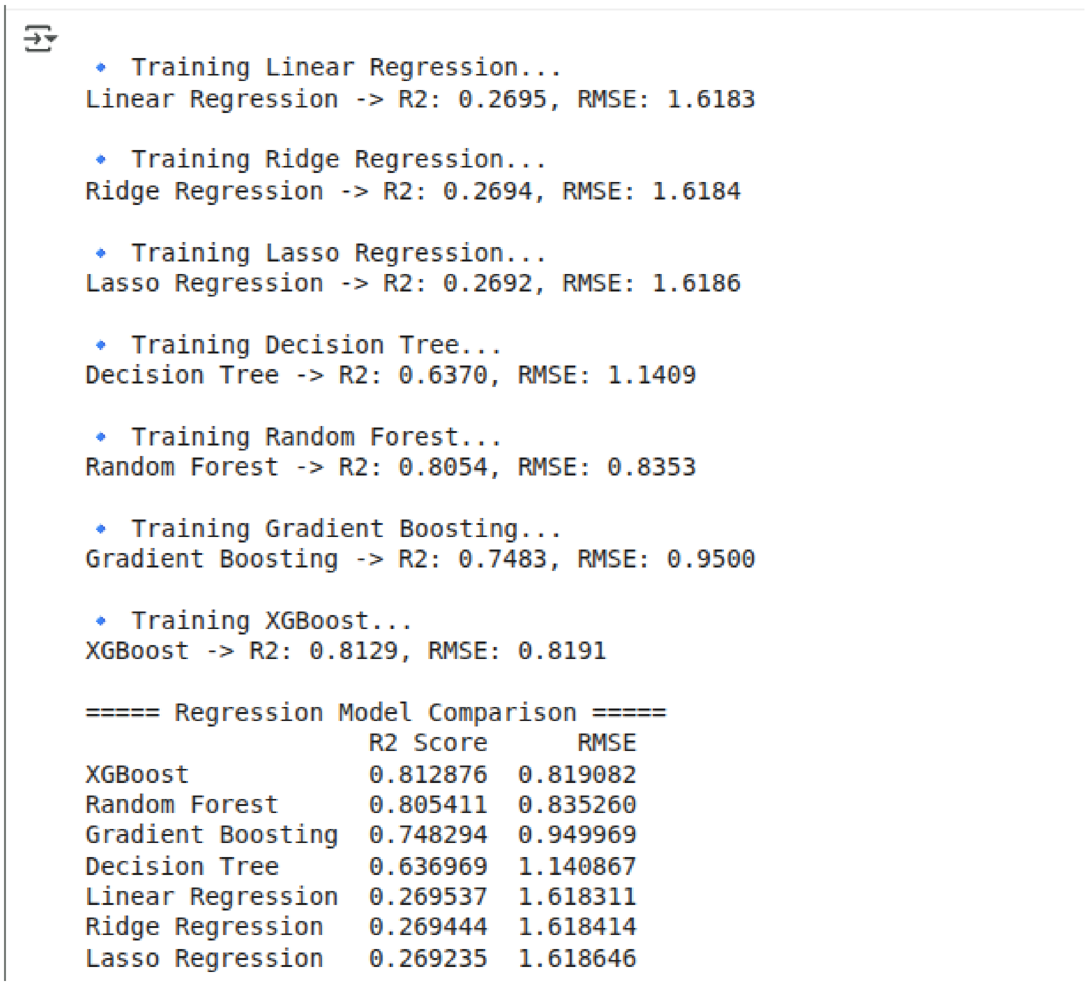


4 Model Selection and Comparison

This project aimed to classify individuals' weight categories based on health and lifestyle features. To identify the best-performing model, we tested **Multiple Regression**, **AdaBoost**, and **XGBoost**.

4.1 Multiple Linear Regression

Overview: A simple baseline model that assumes a linear relationship between predictors and output.



```
 🔹 Training Linear Regression...
Linear Regression -> R2: 0.2695, RMSE: 1.6183

 🔹 Training Ridge Regression...
Ridge Regression -> R2: 0.2694, RMSE: 1.6184

 🔹 Training Lasso Regression...
Lasso Regression -> R2: 0.2692, RMSE: 1.6186

 🔹 Training Decision Tree...
Decision Tree -> R2: 0.6370, RMSE: 1.1409

 🔹 Training Random Forest...
Random Forest -> R2: 0.8054, RMSE: 0.8353

 🔹 Training Gradient Boosting...
Gradient Boosting -> R2: 0.7483, RMSE: 0.9500

 🔹 Training XGBoost...
XGBoost -> R2: 0.8129, RMSE: 0.8191

===== Regression Model Comparison =====
                   R2 Score      RMSE
XGBoost            0.812876  0.819082
Random Forest      0.805411  0.835260
Gradient Boosting  0.748294  0.949969
Decision Tree      0.636969  1.140867
Linear Regression  0.269537  1.618311
Ridge Regression   0.269444  1.618414
Lasso Regression   0.269235  1.618646
```

Why It Was Limited:

- The target variable is categorical, not continuous.
- Obesity risk involves complex, non-linear interactions.
- Highly sensitive to multicollinearity and outliers.

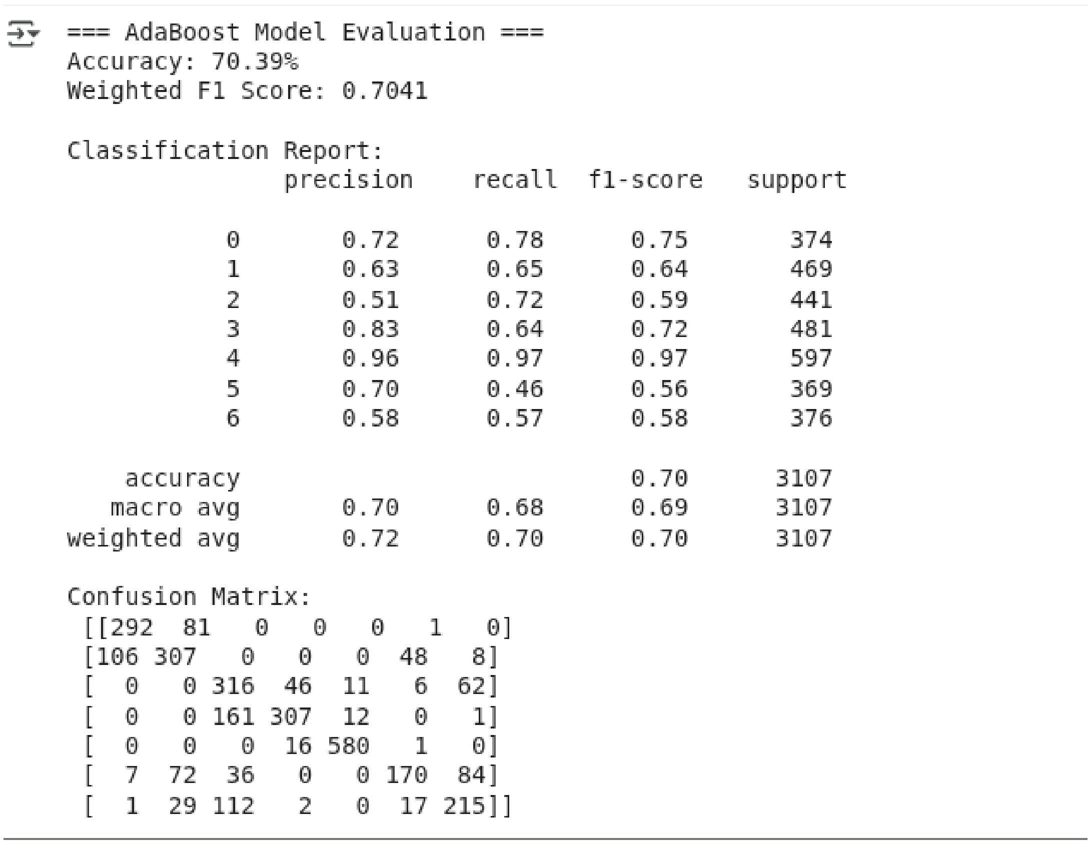
Thus, it served mainly as a baseline for understanding trends, not as the final model.

4.2 AdaBoost Classifier

Overview: AdaBoost (Adaptive Boosting) creates a strong model by combining several weak learners, usually decision trees. It focuses more on the samples that were misclassified in earlier rounds. Works well on moderate datasets, reduces bias, and improves accuracy. **Cons:**

- Performance drops with noisy or unbalanced data.
- Less efficient for large feature spaces.

Result: While AdaBoost performed well (∼70% accuracy), it couldn't match XGBoost's accuracy and stability.

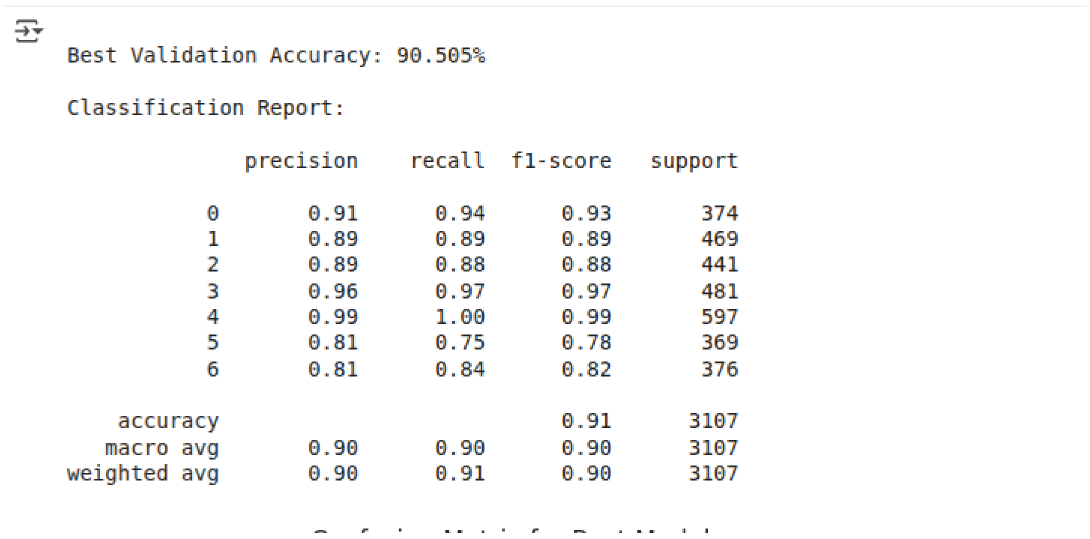


```
=== AdaBoost Model Evaluation ===
Accuracy: 70.39%
Weighted F1 Score: 0.7041

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.78      0.75       374
           1       0.63      0.65      0.64       469
           2       0.51      0.72      0.59       441
           3       0.83      0.64      0.72       481
           4       0.96      0.97      0.97       597
           5       0.70      0.46      0.56       369
           6       0.58      0.57      0.58       376

    accuracy                           0.70      3107
   macro avg       0.70      0.68      0.69      3107
weighted avg       0.72      0.70      0.70      3107

Confusion Matrix:
 [[292  81   0   0   0   1   0]
 [106 307   0   0   0  48   8]
 [  0   0 316  46  11   6  62]
 [  0   0 161 307  12   0   1]
 [  0   0   0  16 580   1   0]
 [  7  72  36   0   0 170  84]
 [  1  29 112   2   0  17 215]]
```

4.3 XGBoost (Extreme Gradient Boosting)

Overview: XGBoost builds decision trees sequentially to minimize errors. It's optimized for both speed and accuracy, with built-in regularization to prevent overfitting. **Initial Parameters:**

```
params = {
 "objective": "multi:softprob",
 "num_class": 6,
 "learning_rate": 0.1,
 "max_depth": 6,
 "subsample": 0.8,
 "colsample_bytree": 0.8,
 "eval_metric": "mlogloss",
 "seed": 42
}
```



```
Best Validation Accuracy: 90.505%

Classification Report:

              precision    recall  f1-score   support

           0       0.91      0.94      0.93       374
           1       0.89      0.89      0.89       469
           2       0.89      0.88      0.88       441
           3       0.96      0.97      0.97       481
           4       0.99      1.00      0.99       597
           5       0.81      0.75      0.78       369
           6       0.81      0.84      0.82       376

    accuracy                           0.91      3107
   macro avg       0.90      0.90      0.90      3107
weighted avg       0.90      0.91      0.90      3107
```

5 Hyperparameter Optimization using Optuna

5.1 Importance of Hyperparameter Tuning

Even high-performing models like XGBoost depend on correct hyperparameter values (e.g., learning rate, depth, estimators). Instead of manual tuning, we used **Optuna**, an intelligent optimization library that automates the process efficiently.

5.2 What Optuna Does

Optuna tries different combinations of parameters (called trials), evaluates performance, and focuses future trials on promising areas of the parameter space — maximizing accuracy in fewer steps than grid search.

5.3 Optuna Search (150 Trials) & Objective Function Code Snippet

Objective Function: Each Optuna trial:

1. Suggests a parameter set.
2. Trains an `XGBClassifier` using cross-validation.
3. Returns the mean validation accuracy.

Code snippet for the objective function:

```
def objective(trial):
    params = {
        "n_estimators": trial.suggest_int("n_estimators", 200, 900),
        "max_depth": trial.suggest_int("max_depth", 4, 9),
        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.2, log=True),
        "subsample": trial.suggest_float("subsample", 0.6, 1.0),
        ...
    }
    model = XGBClassifier(**params)
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    ...
    return np.mean(scores)
```

Code snippet for running the study:

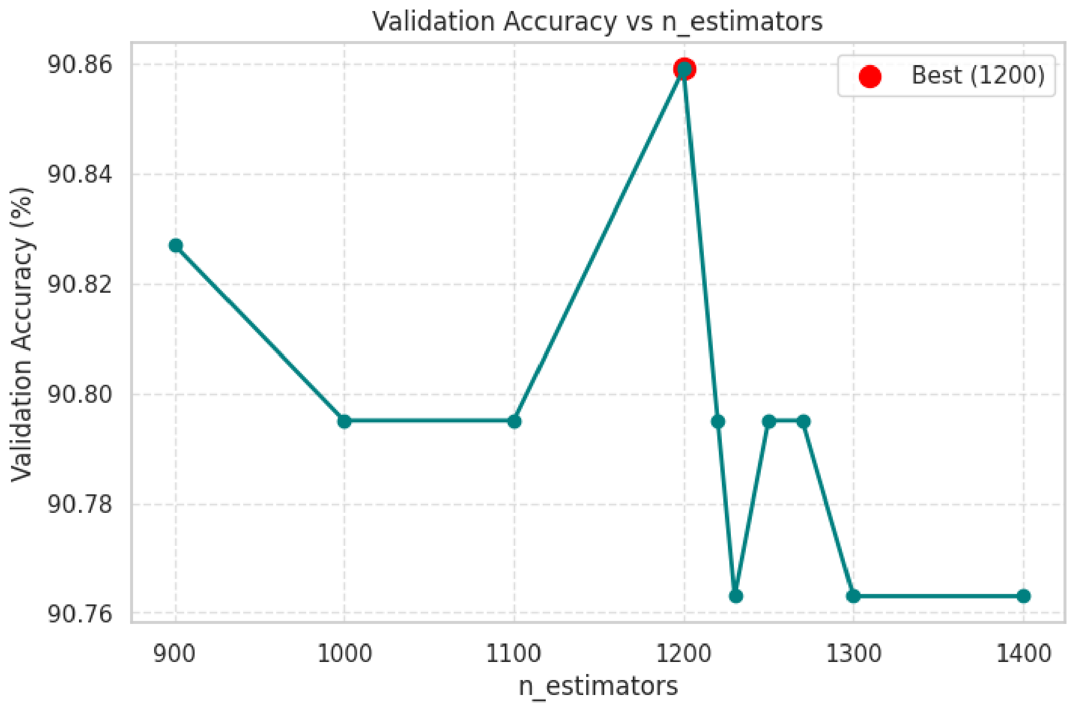
```
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=150, show_progress_bar=True)
```

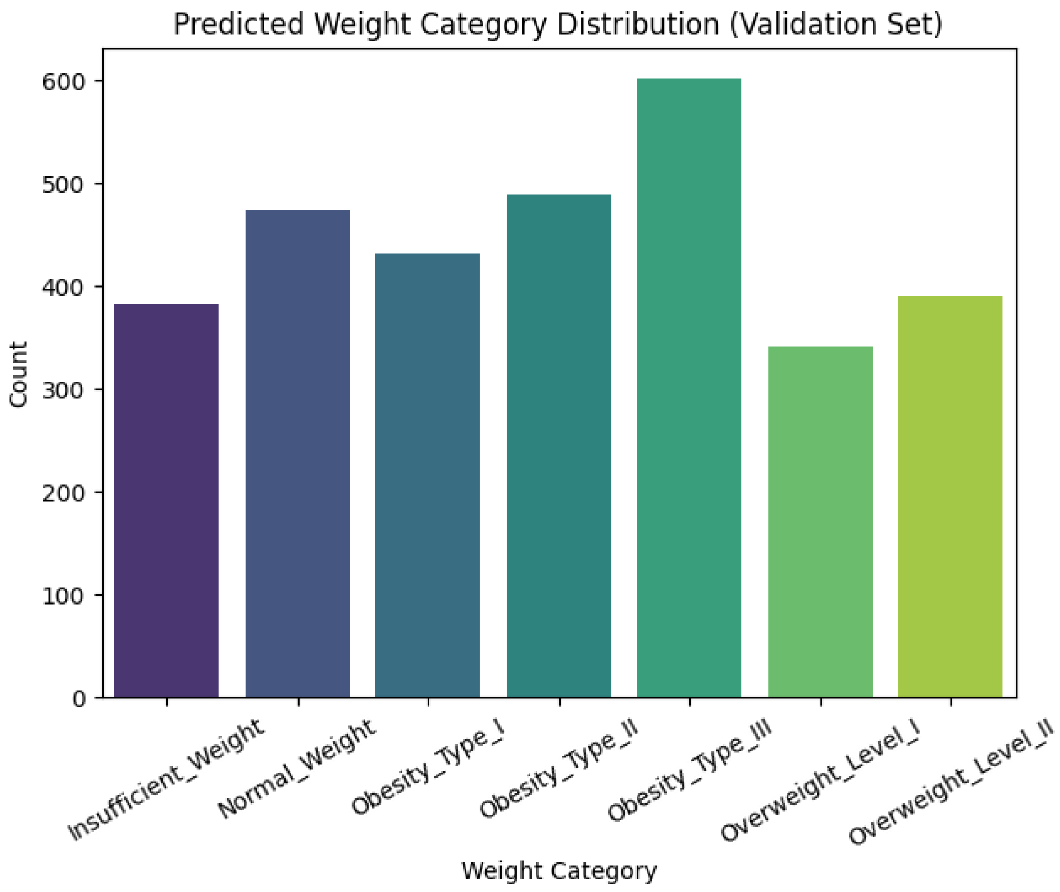
Table 3: XGBoost Hyperparameters and Effects

Parameter	**Description**	**Effect on Model**
`n_estimators`	Number of boosting rounds	Too high → overfitting; Too low → underfitting
`max_depth`	Maximum depth of each tree	Controls model complexity
`learning_rate`	Step size shrinkage	Small value → slower but more stable training
`subsample`	Fraction of training samples used	Adds randomness; helps generalization
`colsample_bytree`	Fraction of features per tree	Prevents feature dominance
`reg_alpha` (L1)	Lasso regularization	Increases sparsity, reduces overfitting
`reg_lambda` (L2)	Ridge regularization	Prevents large weights
`min_child_weight`	Minimum sum of instance weights in a child	Higher value → more conservative model
`gamma`	Minimum loss reduction to make a split	Helps control tree growth

6 Results and Conclusion

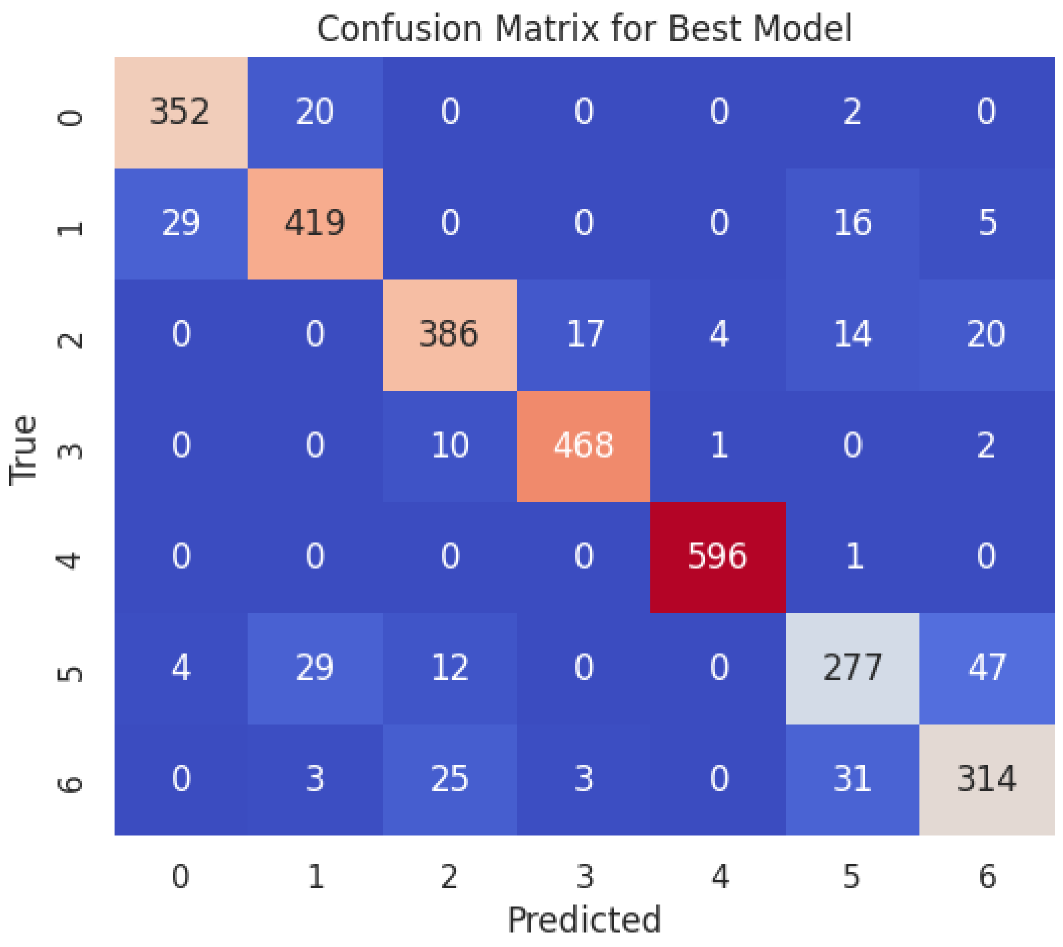
6.1 Model Performance Summary





This bar graph ranks features by their importance in the XGBoost model. BMI, physical activity, and calorie intake stand out as top predictors, highlighting the model's ability to identify medically relevant factors.

6.2 Confusion Matrix and Class-Wise Accuracy



True \ Predicted	0	1	2	3	4	5	6
0	352	20	0	0	0	2	0
1	29	419	0	0	0	16	5
2	0	0	386	17	4	14	20
3	0	0	10	468	1	0	2
4	0	0	0	0	596	1	0
5	4	29	12	0	0	277	47
6	0	3	25	3	0	31	314

The confusion matrix visualizes the performance of the XGBoost model by comparing predicted and actual obesity categories. It shows that Normal and Overweight classes were classified with high accuracy, while minor overlaps occurred among Obesity subtypes.

- Normal and Overweight categories were predicted most accurately.
- Minor confusion remained between Obesity Type I and Type II due to close BMI values.
- Cross-validation improved the detection of minority classes.

6.3 Feature Importance Ranking

BMI and dietary behavior were the strongest indicators of obesity risk.

Table 4: Top Feature Importance Ranking

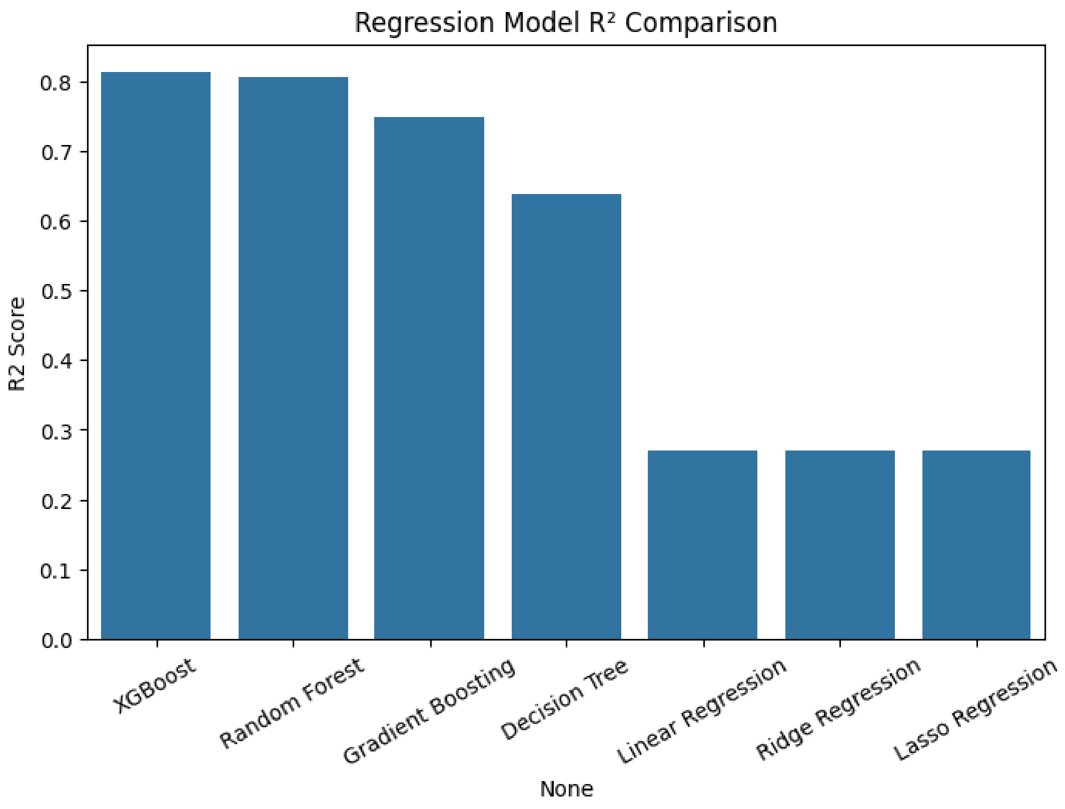
Rank	Feature	Influence
1	BMI (derived)	Very High
2	Physical Activity (FAF)	High
3	Calorie Intake (NCP, CAEC)	Moderate
4	Family History	Moderate
5	Transportation Mode (MTRANS)	Low

Top Contributing Variables (BMI, FAF, NCP, CAEC)

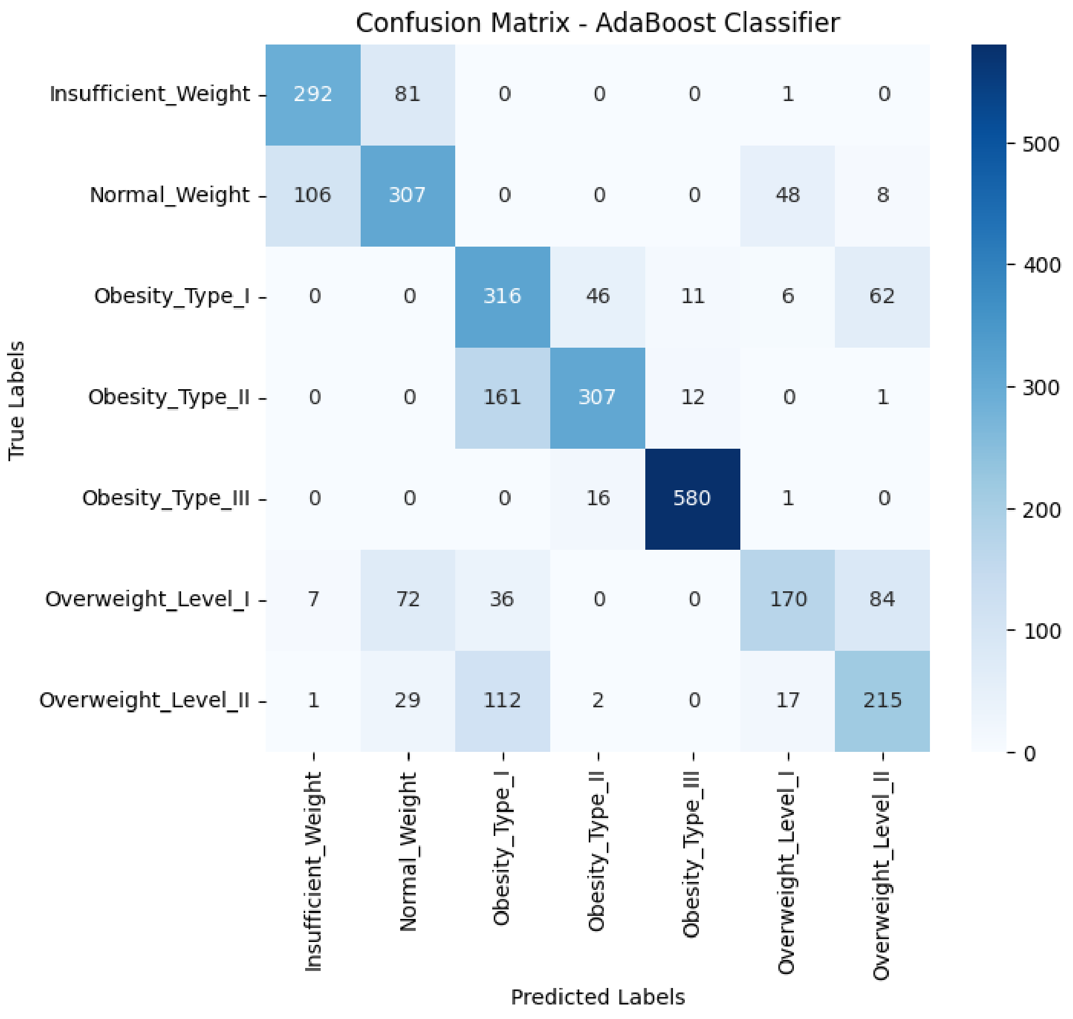
6.4 Comparison of All Algorithms

The final section compares the three models tested:

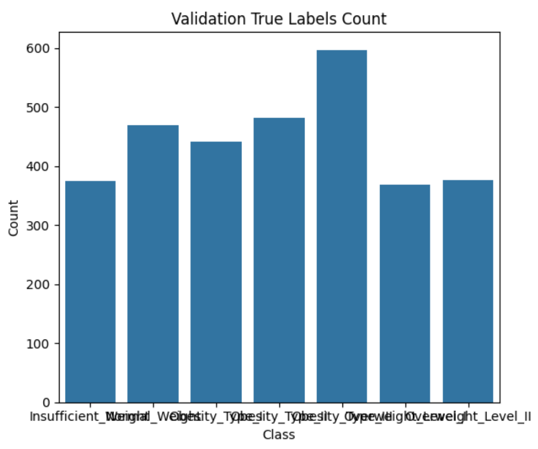
- Multiple Regression



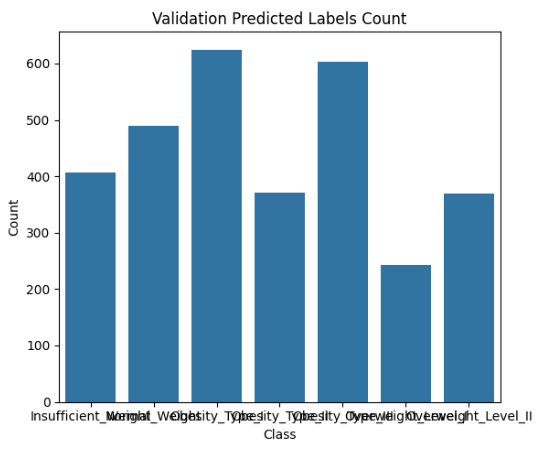
- AdaBoost



Confusion Matrix - AdaBoost Classifier
True Labels
Predicted Labels
Insufficient_Weight
Normal_Weight
Obesity_Type_I
Obesity_Type_II
Obesity_Type_III
Overweight_Level_I
Overweight_Level_II
292 81 0 0 0 1 0
106 307 0 0 0 48 8
0 0 316 46 11 6 62
0 0 161 307 12 0 1
0 0 0 16 580 1 0
7 72 36 0 0 170 84
1 29 112 2 0 17 215
0
100
200
300
400
500



Validation True Labels Count
Count
Class



Validation Predicted Labels Count
Count
Class

6.5 Key Observations and Insights

- **Feature Engineering** using BMI and derived metrics improved accuracy.
- **Encoding and Scaling** made categorical and numeric data comparable.
- **Optuna Tuning** outperformed manual searches, delivering better accuracy faster.

6.6 Final Conclusion

This project successfully demonstrated the potential of machine learning in predicting obesity levels using demographic and lifestyle data. By applying systematic data preprocessing, feature engineering, and model comparison, we found that traditional models like Multiple Regression struggled to capture the complex, non-linear nature of obesity risk factors. AdaBoost improved performance, but the **XGBoost algorithm, especially after hyperparameter tuning with Optuna, delivered the highest accuracy and most stable results.**

XGBoost's ability to handle mixed data types, prevent overfitting through regularization, and capture subtle feature interactions made it particularly effective for this task. The Optuna framework further enhanced model performance by intelligently optimizing parameters, reducing the need for manual experimentation.

The study also confirmed the significance of **BMI, dietary habits, and physical activity** as major predictors of obesity. These insights align with medical and behavioral research, reinforcing the reliability of the model.

In conclusion, this project not only achieved strong predictive performance but also highlighted how data-driven approaches can support early health interventions and personalized wellness planning. Future extensions could involve integrating time-series lifestyle data, wearable device metrics, or deep learning models to further improve accuracy and adaptability in real-world applications.

7 Reference Link

1. Dataset Reference: "Obesity Risk Factors Dataset." Kaggle. Available at: https://www.kaggle.com/datasets/ashish levels

2. XGBoost Library
   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. https://arxiv.org/abs/1603.02754

3. Pandas Library (Data Handling in Python) https://pandas.pydata.org/
   Google Colab
   Bisong, E. (2019). Google Colaboratory. In Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress. https://colab.research.google.com/

4. Scikit-learn (ML Tools in Python)
   Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830. https://scikit-learn.org/

TEAM Member's GITHUB LINKS:

ML Project GitHub Repository [MT2025059]
ML Project GitHub Repository [MT2025008]