Obesity Level Prediction Using Machine Learning

Aman Tiwari (MT2025018)

Mohit Kumar (MT2025075)

Submitted in partial fulfilment for the award of degree of

Master of
Technology In

Computer Science and Engineering

Under The Guidance Of,

Prof. Ashwin Kannan

GitHub Link

Aman: https://github.com/Aman-dps/Machine-Learning

Mohit: https://github.com/mohitkumar-dev95/machine-learning

Index

Abstract:

The dataset used in this study contains information for predicting obesity levels among individuals aged 14 to 61 years. It encompasses a diverse range of personal, behavioral, and physical factors, represented through 17 distinct attributes and consisting of 15,533 data entries. These features capture demographic details, eating habits, physical activity levels, and lifestyle choices — all of which significantly influence an individual's body weight and overall health.

The main goal of this work is to develop an intelligent predictive model that can accurately classify individuals into one of seven predefined weight categories, ranging from Insufficient *Weight to Obesity Type III. This fine-grained classification provides a clearer understanding of* obesity severity. By examining the correlations among input features, the model aims to pinpoint the key factors that most strongly affect obesity levels and use them for precise prediction.

To accomplish this, several machine learning algorithms are considered, including Decision Trees, Random Forests, Support Vector Machines (SVM), and Gradient Boosting models. The performance of these classifiers will be assessed using standard evaluation metrics such as accuracy, precision, recall, and F1-score to determine their predictive capability.

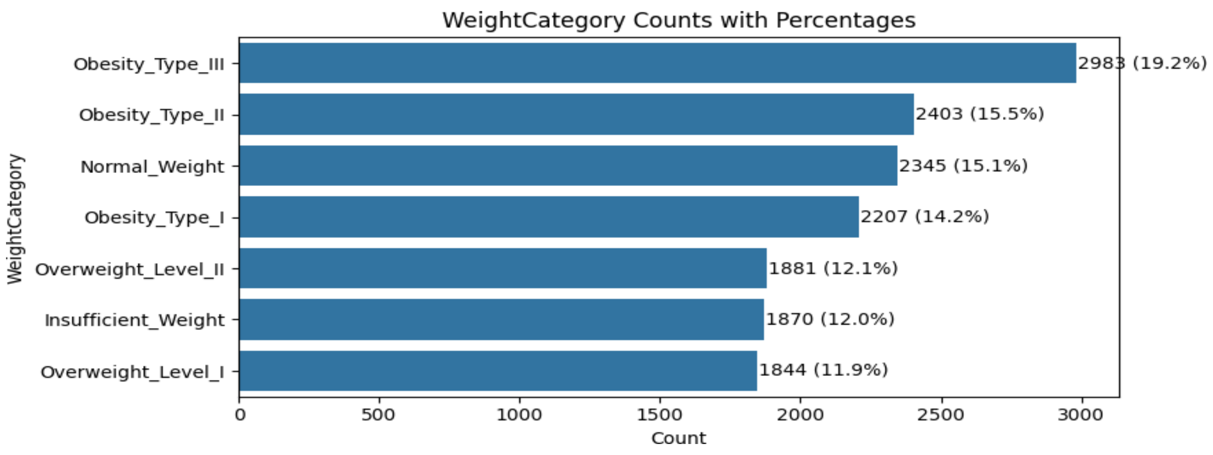
In conclusion, this study aims not only to construct a reliable predictive framework for estimating obesity levels but also to uncover meaningful insights into the behavioral and lifestyle patterns influencing body weight. The outcomes of this research may contribute to developing effective preventive health measures and fostering greater awareness about the impact of daily habits on obesity risk.

Data Exploration:

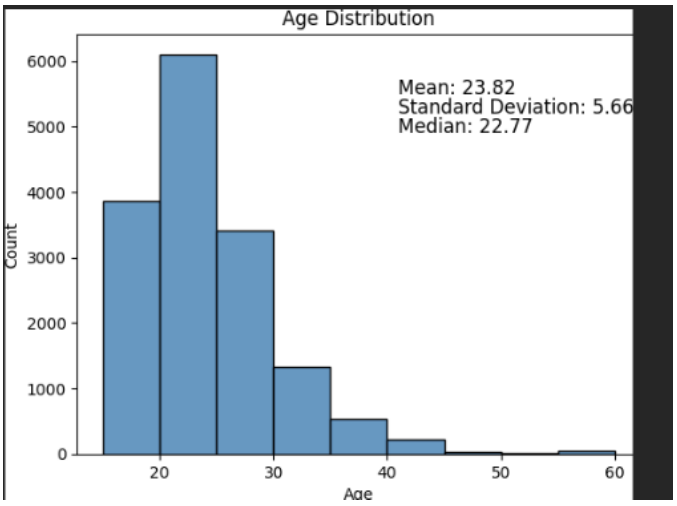
We utilized YData Profiling to perform an initial exploratory analysis of the dataset. This tool provides insights into various aspects such as missing values, duplicate entries, feature correlations, and data imbalance. The analysis revealed that the dataset contained no missing or duplicate records, eliminating the need for any null-value handling or data-cleaning procedures.

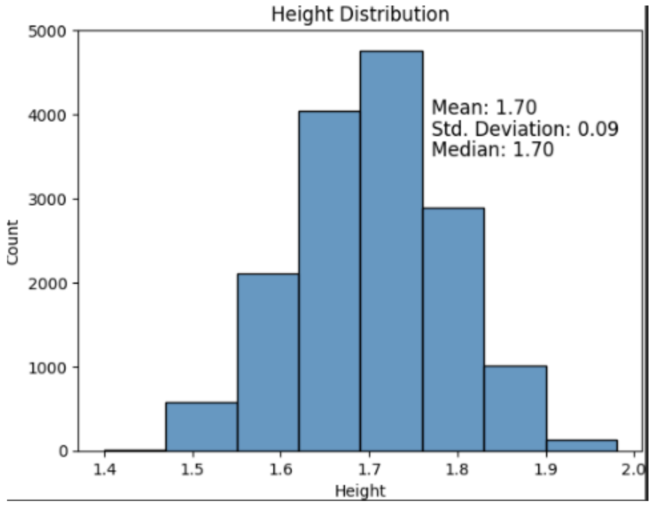
Subsequently, we generated several visualizations to explore feature relationships and identify underlying patterns within the data. Some key observations from this analysis are summarized below:

1. The target variable, Weight_Category, is unevenly distributed across its seven classes. Specifically, the Obesity Type III category has a significantly higher number of records compared to others. This imbalance must be addressed carefully during model training to ensure fair and unbiased learning.

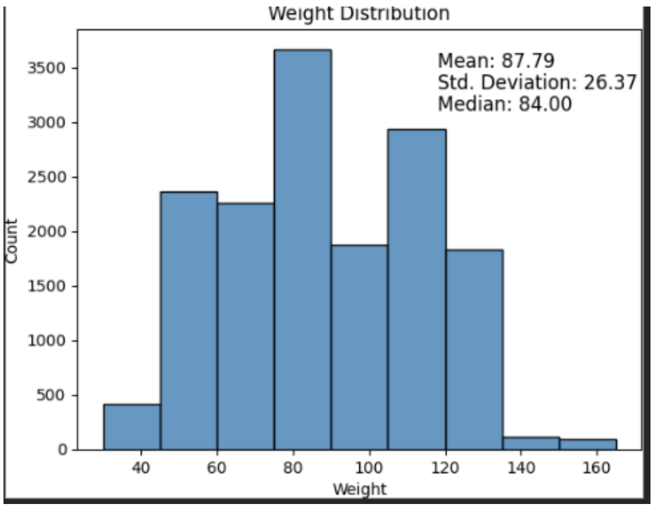


2. The age distribution in the dataset exhibits a slightly skewed bell-shaped pattern, with a clear concentration of individuals between the ages of 20 and 30. This suggests that the dataset is dominated by participants in their twenties. The standard deviation is around 5, which is relatively small considering the full age range. Hence, it is more appropriate to treat age as a **continuous variable rather than dividing it into categorical bins, as such grouping could** di



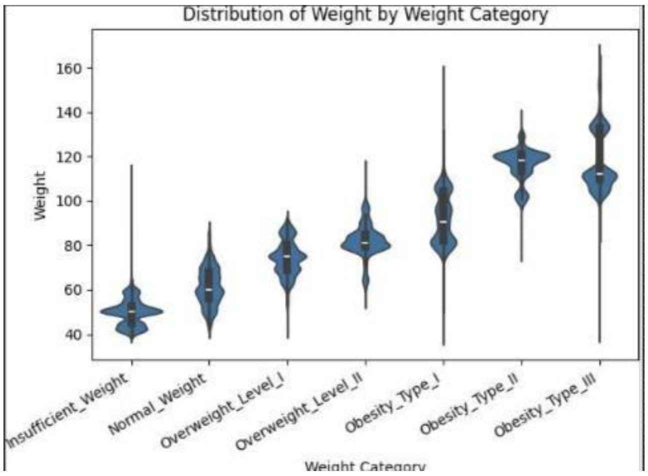
3. The height attribute displays an almost ideal bell-shaped distribution, with both the mean **and median positioned around 1.7. The low standard deviation indicates that most individuals** have heights clustered closely around the average value. As a result, height does not seem to be a highly discriminative feature in differentiating between the various weight categories within the dataset.
4. 

5. The weight feature exhibits a more complex distribution, forming a flatter bell-shaped curve with relatively minor variations in frequency across different ranges. The high standard **deviation indicates a wide spread and substantial variability in individuals' weights within the** dataset. It would be valuable to analyze the relationship between this feature and the target **var**



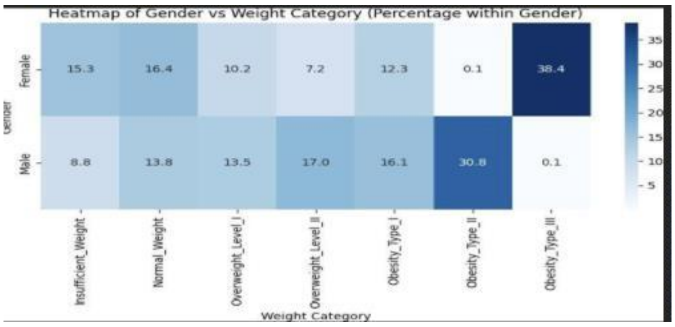
With each weight category, the average weight seems to go higher with more datapoints around the average for each category. This implies a correlation between weight and weight category.

So the below image shows that they have high correlation

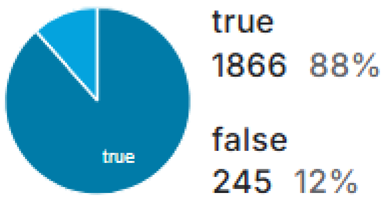


When examining the relationship between gender and weight category through a heatmap, we notice that the distribution for females is more skewed toward the extreme categories, while that of males is relatively centered. Specifically, around 38.4% of females belong to the

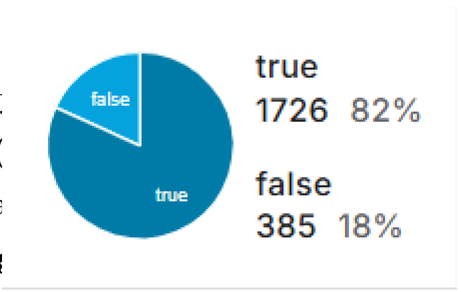
O



5. The FAVC (Frequent Consumption of High-Caloric Food) feature is highly imbalanced, with nearly 91% of the entries marked as "Yes." However, an important insight emerges: when the value is "No," the corresponding weight categories rarely fall within the obese classes. Therefore, despite the class imbalance, this attribute holds strong predictive significance, effectively helping to differentiate between obese and non-obese individuals.

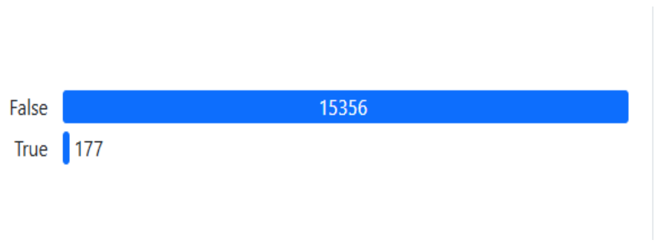


6. The family history feature also displays an imbalanced distribution but exhibits a **strong association with the weight category. Individuals without a family history of obesity are primarily concentrated in the Insufficient Weight, Normal Weight,** and Overweight groups. Conversely, those with a family history of obesity tend to appear more frequently within the obese categories, indicating a clear hereditary influence on obesity risk.



Based on the above anal
Gender, Weight, FAVC (
Consumption of High-Ca
History of Obesity emer
influential factors in determining an individual's weight category. These variables exhibit strong relationships with obesity levels, making them highly valuable predictors for the c

Other attributes in the dataset — such as the Number of Main Meals, Consumption of **Food Between Meals, Smoking Habits, and Mode of Transportation — provide additional** contextual information but demonstrate a relatively weaker direct impact on the target variable. For instance, the Smoking feature shows minimal importance, as reflected in the analysis presented below.



To obtain a comprehensive overview of the dataset, we employed the ydata_profiling library (previously known as pandas-profiling). This tool automatically generates an in-depth **Exploratory Data Analysis (EDA) report, offering detailed insights into each feature,** including distributions, correlations, missing values, and potential data imbalances.

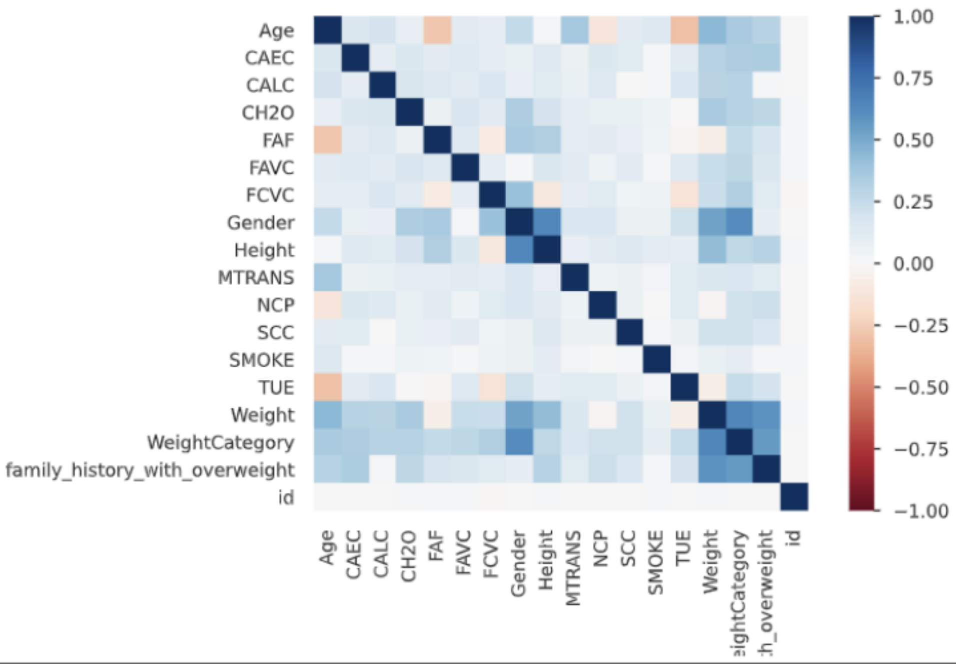
from ydata_profiling import ProfileReport

profile = ProfileReport(train_df, title='Pandas Profiling Report', explorative=True)

profile.to_notebook_iframe()

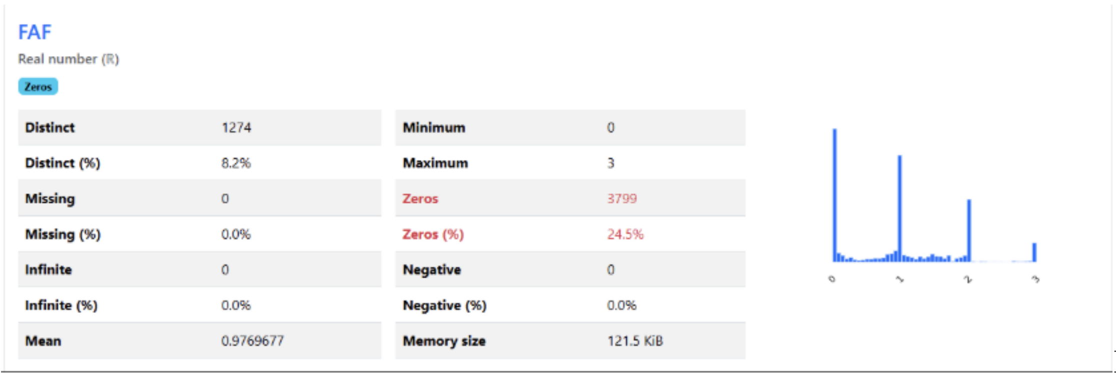
- In the above code snippet, the ProfileReport function is applied to the training dataset **train_df, producing a comprehensive analytical report titled "Pandas Profiling Report." The parameter explorative=True enables an extended exploration mode,** allowing for deeper data profiling and interactive visualizations within the notebook interface.Setting explorative=True enables a more advanced version of the report, which includes interactive visualizations and deeper statistical insights.

  The to_notebook_iframe() method enables the report to be displayed directly within a **Jupyter Notebook, providing an interactive environment for data exploration. This makes it** easy to analyze summary statistics, feature correlations, missing values, and data **distributions without leaving the notebook, thereby streamlining the exploratory data analysis** pr



Data_Set Preparation For Modelling:

Before moving on to the modeling phase, it is crucial to develop a clear understanding of the **structure, quality, and distribution of the dataset. To accomplish this, we conducted an** extensive exploratory data analysis (EDA) using the ydata_profiling library. This tool automatically produces a comprehensive report that includes detailed statistical summaries, **feature distributions, correlation matrices, missing value analysis, and other key insights** for every attribute in the dataset, helping to identify patterns and potential data issues early in the process. Profiling looks like this for each feature ..


FAF

Real number (ℝ)

Zeros

Distinct	1274	**Minimum**	0
Distinct (%)	8.2%	**Maximum**	3
Missing	0	**Zeros**	3799
Missing (%)	0.0%	**Zeros (%)**	24.5%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.9769677	**Memory size**	121.5 KiB

From this we get to know about distinct, missing, mean and other related information about the feature

num

We can change the other features to numerical using one hot encoding

```
transformer = ColumnTransformer(transformers=[
('t1',OneHotEncoder(drop='first'),['CAEC']),
('t2',OneHotEncoder(drop='first'),['CALC']),
('t3', StandardScaler(), numeric_features),
('t4',OneHotEncoder(drop='first'),['Gender','family_history_with_overweight','FAVC','SMOKE','SCC','MTRANS'])],remai
nder='passthrough')
```

This approach ensures that:

7

- The model can interpret categorical variables numerically, enabling algorithms that require numerical input to process them effectively.
- No artificial or unintended ordinal relationships are introduced between categories — for example, "Male" and "Female" are treated as distinct and independent classes, rather than implying any form of numerical or ordered relationship.

1. All unique categories are retained, ensuring that the model can learn from the **complete range of categorical information present in the dataset and capture all** relevant patterns during training.
2. By integrating the insights gained from the profiling report with suitable **preprocessing techniques such as One-Hot Encoding, we ensure that the dataset is clean, properly structured, and well-prepared for building effective and reliable machine learning models. Reasons for using One-Hot Encoding:**

Avoiding Ordinal Bias:
Label Encoding assigns arbitrary numerical values to categorical features (e.g., Male = 0, *Female = 1), which can unintentionally introduce an ordinal relationship—suggesting that* one category is "greater" or "lesser" than another. This is not meaningful for nominal **variables such as gender or food type. One-Hot Encoding eliminates this issue by creating individual binary columns for each category, ensuring they are treated as independent and** unordered.

Preserving Information:
One-Hot Encoding explicitly represents all unique categories, allowing the model to learn

patterns and relationships associated with each category separately, without any loss of categorical information.

Improved Model Performance:
Many machine learning algorithms, such as tree-based models and neural networks, perform better with One-Hot Encoded features. This is because it prevents the model from misinterpreting categorical variables as ordinal or continuous, leading to more accurate and unbiased learning.

1. **Model Building and Evaluation**

2. **Random Forest**

3. We employed the Random Forest algorithm, a powerful ensemble learning **technique, to classify individuals into different weight categories. Random Forest** constructs multiple decision trees and aggregates their predictions to enhance overall **accuracy and reduce overfitting. Its capability to handle both numerical and categorical variables made it well-suited for this study.**
   After training and evaluating the model, we obtained an accuracy of 89.2%, demonstrating that Random Forest effectively captures complex patterns and relationships within the dataset.

4. **Decision Tree**

   A Decision Tree classifier was also implemented for comparison. This model works by recursively splitting the dataset based on feature values, forming a tree-like structure that is intuitive and interpretable. Although simpler and more prone to overfitting than ensemble methods, it still captures significant trends in the data.
   Upon evaluation, the Decision Tree model achieved an accuracy of 84.9%, indicating that it delivers a reasonable predictive performance while providing clear insights into feature importance.

5. XGBoost (Default)

We utilized XGBoost, a highly efficient gradient boosting algorithm, to predict the weight categories. XGBoost builds an ensemble of decision trees sequentially, optimizing for the errors of previous trees while incorporating regularization to prevent overfitting. Its speed, scalability, and ability to handle both numerical and categorical features make it well-suited for our dataset.
After training and evaluation, the default XGBoost model achieved an accuracy of 90.3%, outperforming the other models. This confirms that gradient boosting effectively captures **complex patterns and relationships in the data.**

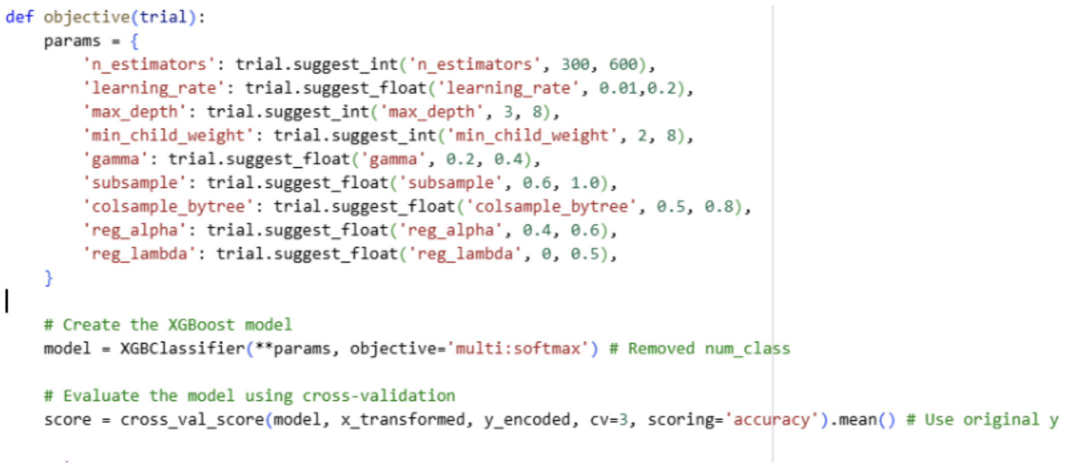
6. XGBoost (Tuned)

To further enhance performance, we performed hyperparameter tuning on XGBoost. By adjusting parameters such as learning rate, tree depth, and number of **estimators, the model was able to better capture the underlying patterns in the** dataset.
Following tuning and evaluation, the XGBoost model achieved an accuracy of **91.65%, surpassing all other models tested. This demonstrates that careful hyperparameter optimization can significantly improve predictive performance,** making the tuned XGBoost model the most effective approach for classifying weight categories in this dataset.

Model	**Accuracy (%)**
Decision Tree	84.9
Random Forest	89.2
AdaBoost	87.3
XGBoost	90.3
Tuned XGBoost	91.65

Parameter Tuning For XG_Boost:

First Tuned Model



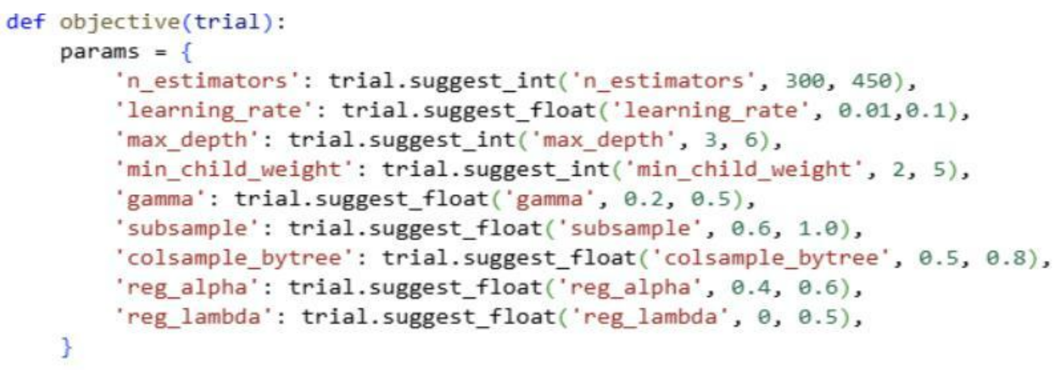
```
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 300, 600),
        'learning_rate': trial.suggest_float('learning_rate', 0.01,0.2),
        'max_depth': trial.suggest_int('max_depth', 3, 8),
        'min_child_weight': trial.suggest_int('min_child_weight', 2, 8),
        'gamma': trial.suggest_float('gamma', 0.2, 0.4),
        'subsample': trial.suggest_float('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 0.8),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.4, 0.6),
        'reg_lambda': trial.suggest_float('reg_lambda', 0, 0.5),
    }

    # Create the XGBoost model
    model = XGBClassifier(**params, objective='multi:softmax') # Removed num_class

    # Evaluate the model using cross-validation
    score = cross_val_score(model, x_transformed, y_encoded, cv=3, scoring='accuracy').mean() # Use original y
```

Accuracy :90.9090%

. . . .

By narrowing and fine-tuning the hyperparameter ranges—particularly n_estimators, **learning_rate, min_weight, min_child_weight, and max_depth—the tuned XGBoost** model achieved better generalization and higher accuracy. These incremental adjustments helped reduce overfitting and enabled the model to focus on the most effective tree **structures, capturing patterns more accurately while maintaining robustness on unseen data.**

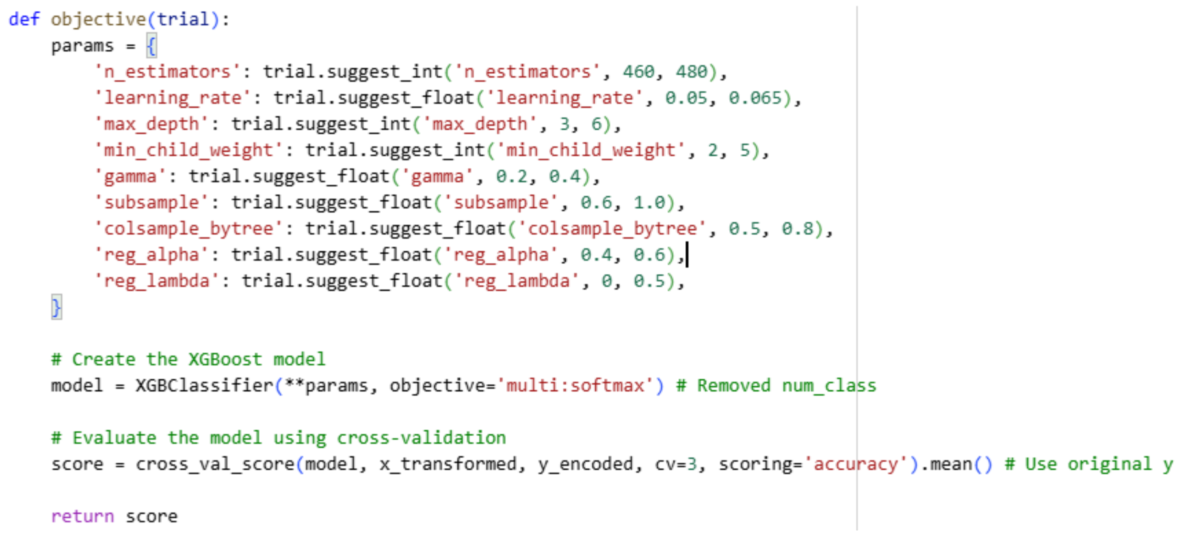
Fourth Tuned Model



```
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 300, 450),
        'learning_rate': trial.suggest_float('learning_rate', 0.01,0.1),
        'max_depth': trial.suggest_int('max_depth', 3, 6),
        'min_child_weight': trial.suggest_int('min_child_weight', 2, 5),
        'gamma': trial.suggest_float('gamma', 0.2, 0.5),
        'subsample': trial.suggest_float('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 0.8),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.4, 0.6),
        'reg_lambda': trial.suggest_float('reg_lambda', 0, 0.5),
    }
```

Accuracy :91.377%

Final Tuned Model:



```
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 460, 480),
        'learning_rate': trial.suggest_float('learning_rate', 0.05, 0.065),
        'max_depth': trial.suggest_int('max_depth', 3, 6),
        'min_child_weight': trial.suggest_int('min_child_weight', 2, 5),
        'gamma': trial.suggest_float('gamma', 0.2, 0.4),
        'subsample': trial.suggest_float('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 0.8),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.4, 0.6),
        'reg_lambda': trial.suggest_float('reg_lambda', 0, 0.5),
    }

    # Create the XGBoost model
    model = XGBClassifier(**params, objective='multi:softmax') # Removed num_class

    # Evaluate the model using cross-validation
    score = cross_val_score(model, x_transformed, y_encoded, cv=3, scoring='accuracy').mean() # Use original y

    return score
```

Accuracy :91.652%

After completing all tuning steps and experimental trials, the final set of hyperparameters for XGBoost achieved the highest accuracy on this dataset. This indicates that, for this specific dataset, these tuned parameters provide the optimal configuration, outperforming all previous model settings and ensuring the most effective predictive performance.

Conclusion:

In this project, I gained practical experience in several key aspects of machine learning, including encoding labels, preprocessing and cleaning data, and analyzing the performance of different models such as Decision Trees, Random Forests, AdaBoost, and XGBoost. I also explored hyperparameter tuning using Optuna, learning how parameters like n_estimators, learning_rate, max_depth, min_child_weight, and gamma affect model performance.

Through systematic experimentation and tuning, I identified the optimal parameters for XGBoost, achieving the highest accuracy on the dataset. This project reinforced the
i

robust and accurate predictive models, and strengthened my understanding of the end-to-end machine learning workflow.