Report on AIT511: Course Project 1 - Multiclass Obesity Prediction

Prateek Kumar Yadav

October 26, 2025

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 Multiclass Obesity Prediction

Multiclass obesity prediction involves the development of predictive models to classify individuals into different obesity categories based on various features such as demographics, lifestyle factors, and health indicators. Obesity is a complex condition influenced by a multitude of factors, including genetics, diet, physical activity, and environmental factors.

2.1 Objective

The primary objective of multiclass obesity prediction is to accurately classify individuals into different obesity categories, which typically include classes such as underweight, normal weight, overweight, and obese. This classification assists in identifying individuals at risk of obesity-related health issues and allows for targeted interventions and healthcare management strategies.

2.2 Data Collection and Features

Data for multiclass obesity prediction typically include a wide range of features that may influence an individual's obesity status. These features can include demographic information (age, gender, ethnicity), lifestyle factors (dietary habits, physical activity levels), medical history (family history of obesity, presence of comorbidities), and physiological measurements (BMI, body fat percentage, blood pressure).

2.3 Model Development

Machine learning algorithms such as logistic regression, decision trees, random forests, support vector machines, and neural networks are commonly used for developing multiclass obesity prediction models. Feature selection, data preprocessing, and model evaluation techniques play crucial roles in building accurate and robust predictive models.

2.4 Model Evaluation

Evaluation of multiclass obesity prediction models involves assessing their performance in terms of classification accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC-AUC). Cross-validation techniques and confusion matrices are often employed to validate model performance and identify areas for improvement.

2.5 Applications

Multiclass obesity prediction models find applications in various domains, including healthcare, public health policy, and personalized medicine. They can assist healthcare providers in early identification of individuals at risk of obesity-related complications, guiding preventive measures, and designing tailored interventions to promote healthy lifestyles and weight management.

2.6 Challenges and Considerations

Developing accurate multiclass obesity prediction models faces several challenges, including data quality issues, class imbalance, feature selection, and model interpretability. Ethical considerations regarding data privacy and potential biases in predictive algorithms also need to be addressed to ensure fair and equitable healthcare outcomes for all individuals.

In summary, multiclass obesity prediction plays a vital role in addressing the global burden of obesity and its associated health risks by leveraging data-driven approaches to identify at-risk individuals and inform targeted interventions for prevention and management.

3 Introduction

This report details the methodology employed to predict obesity risk levels based on a given dataset. The primary goal was to construct a machine learning model capable of classifying individuals into different obesity categories using a set of lifestyle, dietary, and physical attributes. The approach encompassed data preprocessing, model selection, hyperparameter optimization, and performance evaluation.

Goal: Predict obesity risk in individuals.

NObeyesdad (Target Variable): The target variable consists of the following categories, typically defined by Body Mass Index (BMI) ranges:

- `Insufficient_Weight`: BMI less than 18.5
- `Normal_Weight`: BMI 18.5 to 24.9
- `Overweight_Level_I`, `Overweight_Level_II`: BMI 25.0 to 29.9 (split into two levels)
- `Obesity_Type_I`: BMI 30.0 to 34.9
- `Obesity_Type_II`: BMI 35.0 to 39.9
- `Obesity_Type_III`: BMI Higher than 40

4 Data Processing

The analysis began with processing the `train.csv` dataset to prepare it for modeling. The following steps were performed:

4.1 Loading and Initial Inspection

The dataset was loaded into a pandas DataFrame. An initial inspection using `train_data.head()` was done to understand its structure and content. The original target column is named `NObeyesdad`, which was later renamed/encoded to `WeightCategory` for modeling purposes.

4.2 Data Cleaning

The dataset was checked for quality issues:

- **Missing Values:** Checked using `train_data.isnull().sum()`. No missing values were found.
- **Duplicate Values:** Checked using `train_data.duplicated().sum()`. No duplicate entries were found.

4.3 Exploratory Data Analysis (EDA)

A preliminary visualization of the target variable (`WeightCategory`) was conducted using a count plot generated with Seaborn. This helped understand the distribution of different obesity levels and identify potential class imbalances. The distribution is shown in Figure 1.

Figure 1: Distribution of the target variable 'WeightCategory'

Figure 1: Distribution of the target variable 'WeightCategory'. Shows counts for each obesity level, with Obesity_Type_III having the highest count.

4.4 Feature Encoding and Scaling

To prepare the data for machine learning models, transformations were applied:

- **Target Variable Encoding:** The categorical target variable ('WeightCategory' or 'NObeyesdad') was converted into numerical labels using `sklearn.preprocessing.LabelEncoder`.
- **Feature Splitting:** The dataset was divided into features (X) and the target variable (y). The 'id' column was dropped from the features used for training.
- **Numerical and Categorical Feature Identification:** Features were explicitly separated.
  - Numerical: `['Age', 'Height', 'Weight', 'FCVC', 'NCP', 'CH2O', 'FAF', 'TUE']`
  - Categorical: `['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC', 'SMOKE', 'SCC', 'CALC', 'MTRANS']`
- **Preprocessing Pipeline:** A `ColumnTransformer` was constructed. Numerical features were scaled using `StandardScaler`, and categorical features were converted using `OneHotEncoder`. The `handle_unknown='ignore'` option was used for the OneHotEncoder to handle potential unseen categories in test data.

4.5 Data Splitting

The feature set (X) and encoded target variable (y) were split into training and validation sets using `train_test_split`. An 80%/20% split was used for training and validation, respectively, with `random_state=42` for reproducibility. The Decision Tree/XGBoost notebook also used stratification based on the target variable during the split.

5 Models Used

5.1 Approach 1: LightGBM Classifier

The Light Gradient Boosting Machine (LGBM) Classifier (`lightgbm.LGBMClassifier`) was selected as the primary predictive model in the first analysis. LGBM is known for its speed, efficiency (particularly with large datasets), and high accuracy on structured data. It uses tree-based learning algorithms within a gradient boosting framework. The model was integrated into a scikit-learn `Pipeline` along with the preprocessor to ensure correct application of preprocessing steps.

5.2 Approach 2: Decision Tree and XGBoost

The second analysis explored two models:

- **Baseline Decision Tree:** A standard `DecisionTreeClassifier` from scikit-learn was used as an initial baseline model.
- **XGBoost Classifier:** An `XGBClassifier` was implemented, another powerful gradient boosting algorithm known for strong performance. Like LGBM, it was used within a scikit-learn `Pipeline` including the preprocessor.

6 Model Performance Comparison

Figure 2 provides a visual comparison of the validation accuracies achieved by the different models tested. It clearly shows the superior performance of the gradient boosting methods (XGBoost and LightGBM) over the baseline Decision Tree.

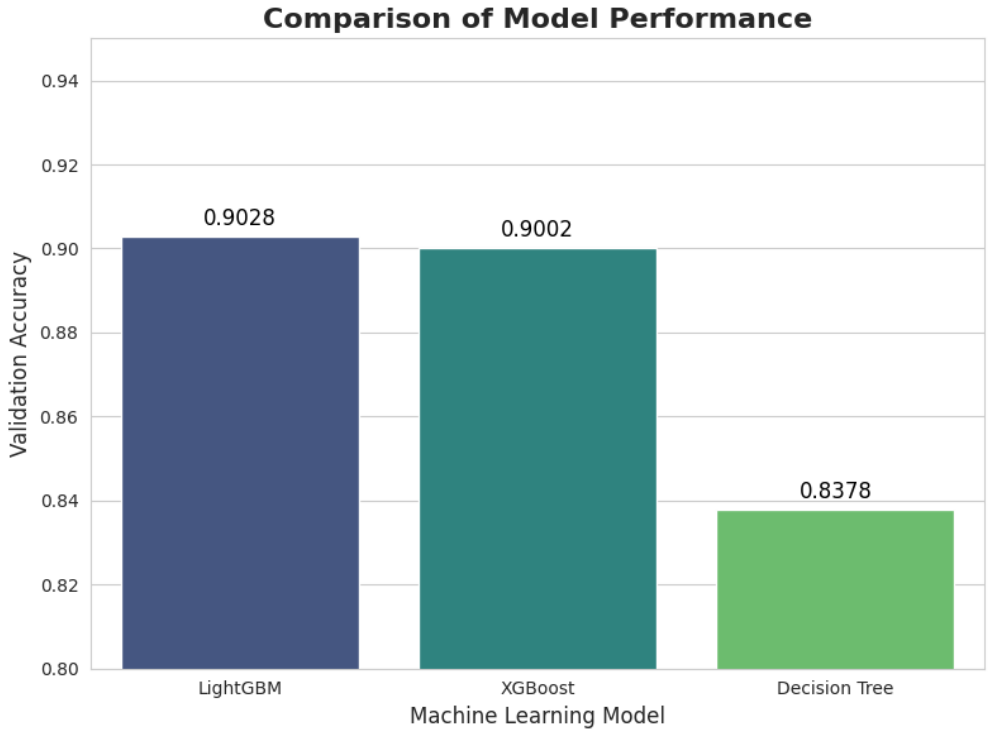

Figure 2: Comparison of Validation Accuracy for Different Models

7 Detailed Visual Analysis

7.1 Age Distribution by Obesity Category

Figure 3 shows the distribution of participant ages, separated by their obesity category. We can observe different age patterns emerging across the groups.

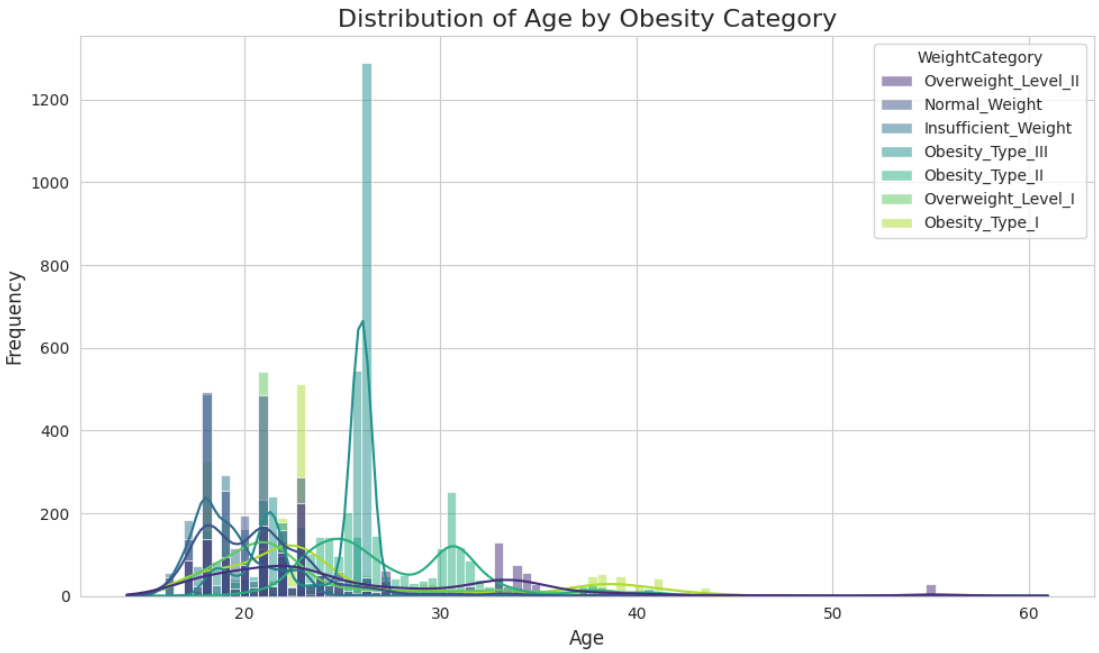

Figure 3: Distribution of Age across different Obesity Categories.

7.2 Weight Distribution by Obesity Category

The box plot in Figure 4 illustrates the spread and central tendency of weight for each obesity category. As expected, there are clear differences in weight distributions across the defined categories.

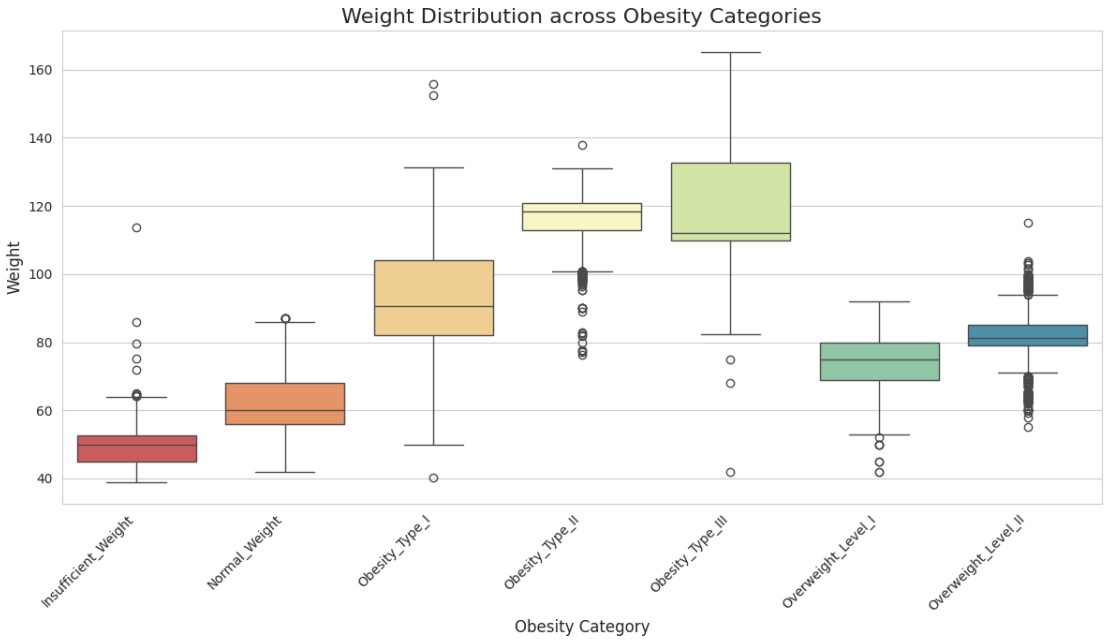

Figure 4: Box Plot of Weight for each Obesity Category.

7.3 High Caloric Food Consumption (FAVC) Analysis

Figure 5 displays the proportion of individuals within each obesity category who frequently consume high-caloric foods (FAVC). This helps visualize the relationship between this dietary habit and weight status.

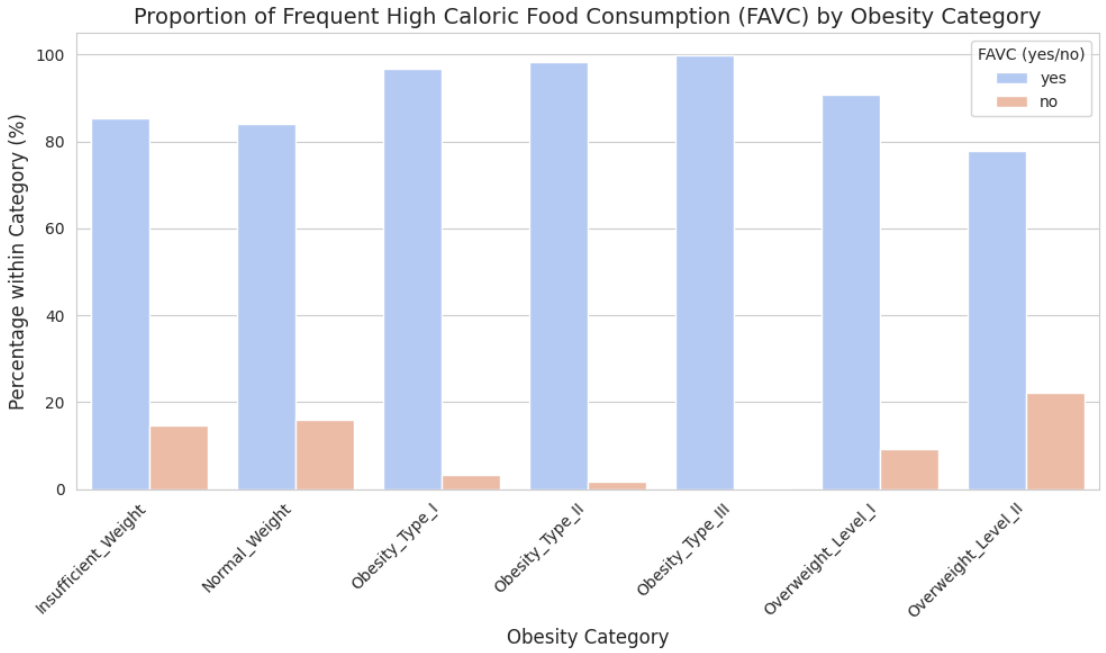

Figure 5: Proportion Bar Chart of FAVC across Obesity Categories.

7.4 Correlation Among Numerical Features

The heatmap in Figure 6 visualizes the pairwise correlations between the numerical features used in the model. This helps identify any strong linear relationships between predictors.

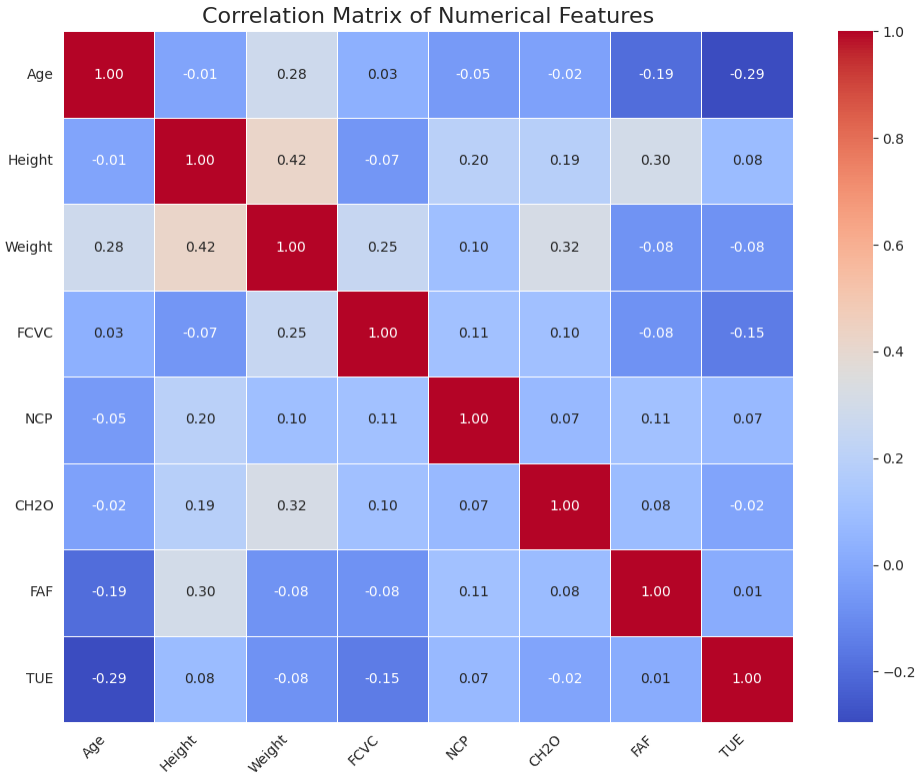

Figure 6: Correlation Heatmap of Numerical Features.

7.5 Model Performance: Confusion Matrix

Figure 7 presents the confusion matrix for the best model's predictions on the validation set. This matrix details the correct and incorrect classifications for each obesity category.

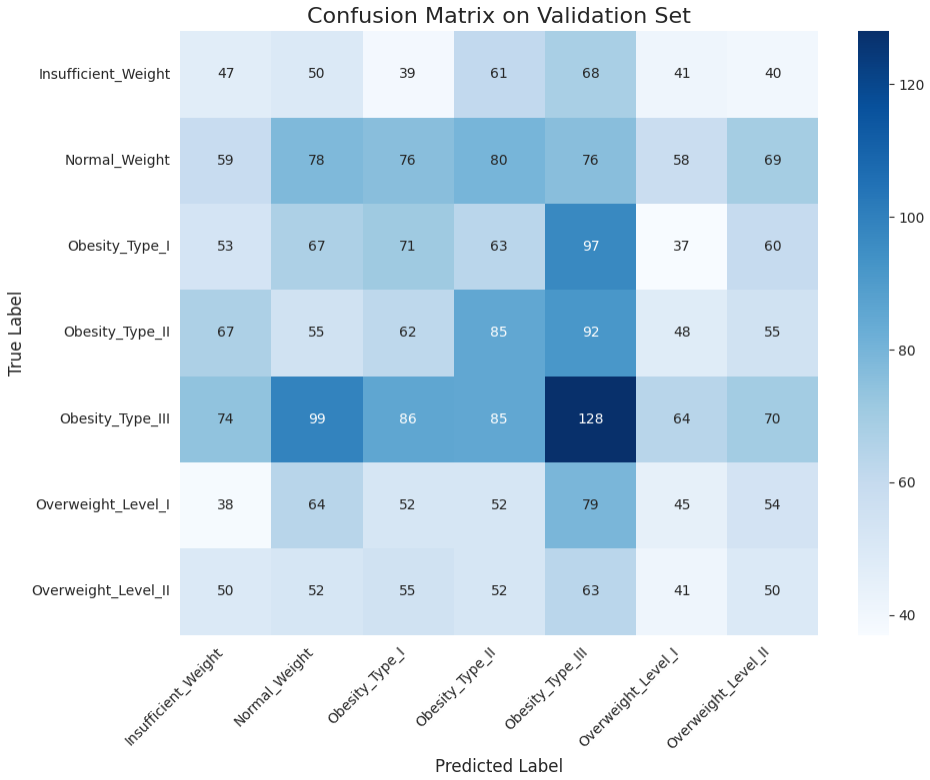

Figure 7: Confusion Matrix for the Decision Tree model on the Validation Set.

8 Hyperparameter Tuning

8.1 Approach 1: LightGBM with RandomizedSearchCV

Hyperparameter tuning for the LGBM model was performed using `sklearn.model_selection.RandomizedSearchCV` to find an optimal combination of parameters.

- **Method:** Randomized Search Cross-Validation.
- **Iterations (`n_iter`):** 120.
- **Cross-Validation (`cv`):** 6 folds.
- **Parallelism (`n_jobs`):** -1 (Utilizes all available CPU cores).

The hyperparameter grid searched was extensive:

```
param_grid = {
    'classifier__n_estimators': [100, 210, 350, 420, 570],
    'classifier__max_depth': [4, 6, 8, 12, 14],
    'classifier__learning_rate': [0.01, 0.03, 0.05, 0.2, 0.3],
    'classifier__subsample': [0.5, 0.7, 0.9, 1.0],
    'classifier__colsample_bytree': [0.5, 0.7, 0.9, 1.0],
    'classifier__min_child_samples': [10, 20, 30, 40, 50],
    'classifier__reg_alpha': [0.0, 0.1, 0.5, 1.0],
    'classifier__reg_lambda': [0.0, 0.1, 0.5, 1.0],
    'classifier__min_child_weight': [1e-3, 1e-2, 0.1, 1, 10]
}
```

Justification for RandomizedSearchCV: For large hyperparameter spaces like the one defined for LGBM, `RandomizedSearchCV` is often preferred over `GridSearchCV`. `GridSearchCV` exhaustively tries every single combination of parameters specified, which can be computationally very expensive and time-consuming. `RandomizedSearchCV`, on the other hand, samples a fixed number (`n_iter`) of parameter combinations from the specified distributions or lists. This allows for a broader exploration of the parameter space within a fixed computational budget and often yields results very close to the optimal ones found by Grid Search, but much faster.

The best parameters found by the randomized search were:

```
Best Parameters: {
 'classifier__subsample': 0.7,
 'classifier__reg_lambda': 0.0,
 'classifier__reg_alpha': 0.0,
 'classifier__n_estimators': 360,
 'classifier__min_child_weight': 0.01,
 'classifier__min_child_samples': 10,
 'classifier__max_depth': 5,
 'classifier__learning_rate': 0.03,
 'classifier__colsample_bytree': 0.5
}
```

8.2 Approach 2: XGBoost with GridSearchCV

Hyperparameter tuning for the XGBoost model used `sklearn.model_selection.GridSearchCV`.

- **Method:** Grid Search Cross-Validation.
- **Cross-Validation (`cv`):** 3 folds.
- **Parallelism (`n_jobs`):** -1.
- **Scoring:** Accuracy.

The parameter grid searched was smaller than for LGBM:

```
param_grid = {
    'classifier__n_estimators': [100, 250],  # Number of trees
    'classifier__max_depth': [3, 5, 7],      # Depth of trees
    'classifier__learning_rate': [0.1]
}
```

The best parameters found by this grid search were:

```
Best parameters found: {
 'classifier__learning_rate': 0.1,
 'classifier__max_depth': 3,
 'classifier__n_estimators': 250
}
```

The best cross-validation accuracy achieved during this search was 90.41%.

9 Results and Discussion

The performance of the different models was evaluated on their respective validation sets (the 20% hold-out portion).

- **Baseline Decision Tree Accuracy:** 83.78%.

- **Baseline XGBoost Accuracy:** 90.02%.
- **Tuned LightGBM Accuracy:** 90.28%.
- **Tuned XGBoost (Best CV Score):** 90.41%. (Note: This is the cross-validation score on the training data, not the final validation score on the hold-out set after refitting with best params).

The baseline Decision Tree performed significantly worse than the gradient boosting methods. Its classification report reveals particular difficulty with the `Overweight_Level_I` and `Overweight_Level_II` classes:

	precision	recall	f1-score	support
Insufficient_Weight	0.86	0.89	0.88	374
Normal_Weight	0.78	0.78	0.78	469
Obesity_Type_I	0.80	0.80	0.80	441
Obesity_Type_II	0.93	0.93	0.93	481
Obesity_Type_III	0.99	1.00	0.99	597
Overweight_Level_I	0.69	0.63	0.66	369
Overweight_Level_II	0.70	0.74	0.72	376
accuracy			0.84	3107
macro avg	0.82	0.82	0.82	3107
weighted avg	0.84	0.84	0.84	3107

Both LightGBM and XGBoost, being gradient boosting algorithms, demonstrated superior performance. These methods build models sequentially, with each new model attempting to correct the errors made by the previous ones, leading to more robust and accurate predictions compared to a single decision tree.

Hyperparameter tuning provided improvements. The baseline XGBoost achieved 90.02%, while the tuned version reached a best cross-validation score of 90.41% on a limited grid. The tuned LightGBM, optimized using Randomized Search over a much larger parameter space, achieved a validation accuracy of 90.28%. This is a strong result, indicating good generalization to unseen data. The performance of tuned XGBoost and tuned LightGBM are very comparable based on these results. The combination of the preprocessing pipeline and hyperparameter tuning was crucial for achieving these high accuracies.

10 Conclusion

This project successfully developed machine learning pipelines for predicting obesity risk. Data preprocessing involved handling categorical and numerical features using 'OneHotEncoder' and 'StandardScaler' within a 'ColumnTransformer'. Baseline models (Decision Tree) showed moderate accuracy, while gradient boosting methods (LightGBM, XGBoost) performed significantly better. Hyperparameter tuning using 'RandomizedSearchCV' for LightGBM and 'GridSearchCV' for XGBoost further refined the models, leading to high validation accuracies around 90.3% - 90.4%. This demonstrates the effectiveness of gradient boosting models, combined with appropriate preprocessing and tuning, for this classification task on structured data. The final best model (LightGBM in the first analysis) was retrained on the entire training dataset to generate predictions for submission.

11 GitHub Repository

The Jupyter notebook containing the complete code for the analysis is available at: `https://github.com/prateekkumaryadav2/SEM_1_Machine_Learning`