Report on Obesity level Estimation Using Decision Tree

Yanko Acuña

Rodrigo Domínguez

Introduction

This report outlines a machine learning approach to estimate obesity levels based on eating habits and physical condition. The dataset used was sourced from the UCI Machine Learning Repository, especially titled “Estimation of Obesity Levels Based on Eating Habits and Physical Condition”. The goal of this project is to create a decision tree model from scratch, without using built-in machine learning libraries like scikit-learn's DecisionTreeClassifier, to classify individuals into different obesity levels.

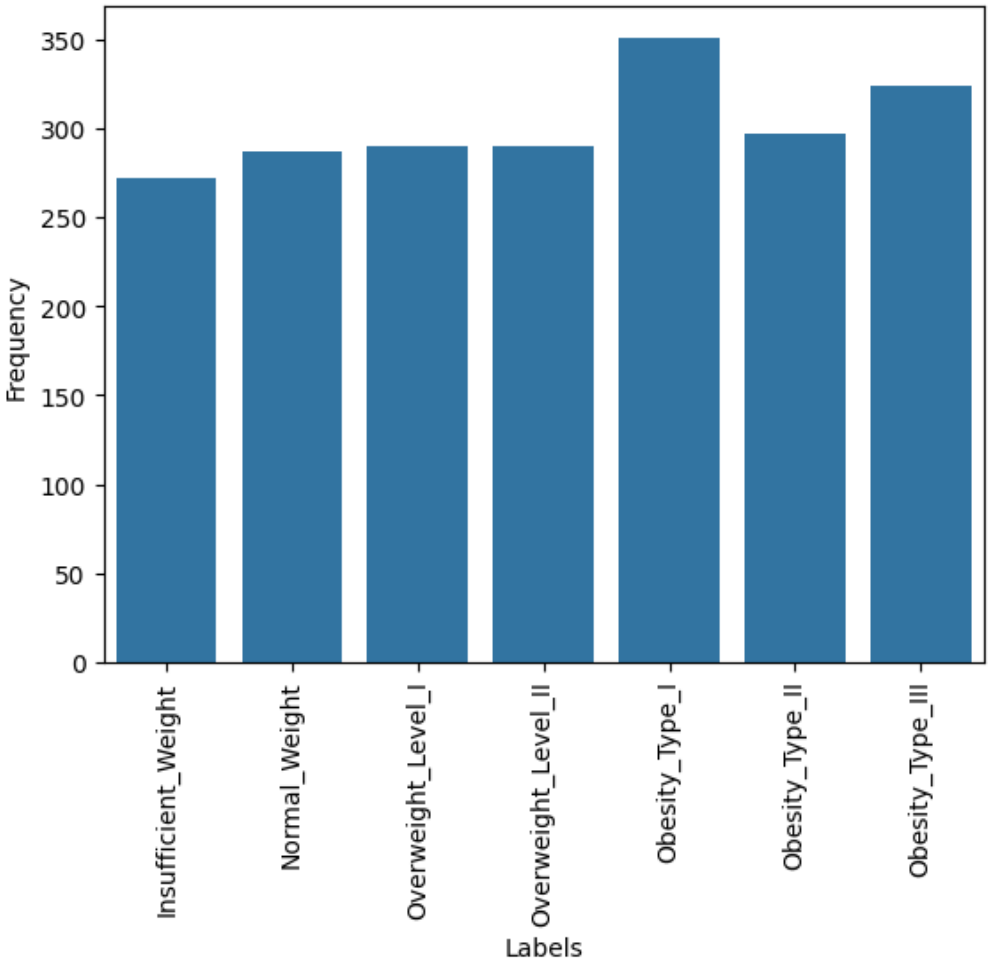
Dataset Description

The dataset contains 2,111 instances and 17 attributes, which include data from the countries of Mexico, Perú and Colombia and include both categorical and numerical data. Each instance describes an individual's demographic information, eating habits, physical activity, and lifestyle choices, and the target variable (NObeyesdad) represents their obesity level. The obesity levels include classes like:

- Insufficient Weight
- Normal Weight
- Overweight Level I & II
- Obesity Types I, II & III

To prepare the data for training, categorical attributes such as Gender, family_history_with_overweight, MTRANS, etc., were label-encoded into numerical values to facilitate processing by the custom decision tree algorithm. 77% of the data was generated synthetically by the creator of the dataset using the Weka tool and the SMOTE filter, 23% of the data was collected directly from users through a web platform.

Distribution of the data:



The dataset includes:

1. **Gender**: The individual's gender (male or female).
2. **Age**: The individual's age (in years).
3. **Height**: The individual's height (in meters).
4. **Weight**: The individual's weight (in kilograms).
5. **family_history_with_overweight**: Indicates if there is a family history of overweight (yes or no).
6. **FAVC (Frequent consumption of high caloric food)**: Indicates if the individual frequently consumes high-calorie food (yes or no).
7. **FCVC (Frequency of consumption of vegetables)**: Frequency of vegetable consumption (on a scale of 1 to 3, with 3 being the highest frequency).
8. **NCP (Number of main meals)**: Number of main meals the individual consumes per day.
9. **CAEC (Consumption of food between meals)**: Frequency of food consumption between meals (never, sometimes, frequently, always).
10. **SMOKE**: Indicates if the individual smokes (yes or no).
11. **CH2O (Daily water intake)**: Amount of water consumed daily (in liters).

12. **SCC (Monitor calorie consumption)**: Indicates if the individual monitors their calorie intake (yes or no).
13. **FAF (Physical activity frequency)**: Frequency of physical activity (in hours per week).
14. **TUE (Time using technology devices)**: Time spent using technological devices (in hours per day).
15. **CALC (Consumption of alcohol)**: Frequency of alcohol consumption (never, sometimes, frequently, always).
16. **MTRANS (Transportation used)**: Type of transportation used by the individual (e.g., walking, bicycle, car, public transportation).
17. **NObeyesdad**: Classification of the individual's obesity level (e.g., Insufficient_Weight, Normal_Weight, Overweight_Level_I, Overweight_Level_II, Obesity_Type_I, Obesity_Type_II, Obesity_Type_III).

Data Preprocessing

In order to utilize the dataset effectively, preprocessing steps were undertaken:

1. **Label Encoding**: Categorical variables such as gender, family history of overweight, and transportation mode were encoded into numerical representations. This is critical for machine learning algorithms to interpret non-numeric data effectively. Label encoding allows for the conversion of qualitative data (e.g., Gender) into numeric values (0 for female, 1 for male, etc.), making it easier to process.
2. **Splitting the Data**: The dataset was split into training and testing subsets with a ratio of 80:20, ensuring a reliable basis for evaluating model performance.

Decision Tree Implementation

A custom decision tree classifier was developed to model the relationship between the attributes and obesity levels. Here are the key aspects of the implementation:

Node Structure:

Each decision node in the tree is implemented as follows:

```
class DecisionNode:
    def __init__(self, feature_index=None, threshold=None, left=None, right=None, value=None):
        self.feature_index = feature_index  # Index of the feature used for splitting
        self.threshold = threshold  # Threshold value for the split
        self.left = left  # Reference to the left child node
        self.right = right  # Reference to the right child node
```

```
        self.value = value  # Value assigned to a leaf node, if
applicable
```

This structure allows each node to represent either an internal decision (with child nodes) or a leaf that contains the final classification with value.

Building tree

Recursive Splitting

The decision tree is built through a process of **recursive splitting**. The goal is to repeatedly divide the dataset into subsets based on the feature that results in the highest information gain. At each stage, the dataset is split according to a **decision rule** on a feature that yields the most homogeneous subsets, helping in categorizing data effectively.

Here is a detailed explanation of how the decision tree is constructed:

1. **Selecting the Best Feature and Threshold**:
    - At each node, the algorithm considers all the features and their possible values (thresholds) to decide on the best way to split the dataset.
    - For each feature, it calculates the information gain that results from a split. The information gain measures how well a split improves the homogeneity of the target variable within the resulting subsets.
    - The split with the highest information gain is chosen as the best split for that node.
2. **Information Gain Calculation**:
    - The information gain is computed as the difference between the parent node's entropy and the weighted average entropy of the child nodes.
    - Entropy is a measure of uncertainty or impurity in the data. A lower entropy indicates a more homogeneous subset. The goal is to reduce entropy as much as possible with each split.
    - The formula for entropy is:

$$Entropy(S) = -\sum_{i=1}^{n} p_i * log_2(p_i)$$

      where p_i is the probability of class i in the subset.

3. **Stopping Criteria**: The recursive splitting continues until one or more of the following conditions are met:
    - **Pure Node**: All samples in the node belong to the same class (i.e., there is no further need to split because there is no uncertainty left).

- **Maximum Depth Reached (max_depth)**: This is a user-defined parameter that limits how deep the tree can grow. A deeper tree can learn more complex patterns but may also overfit the data.
- **Minimum Number of Samples Required for Split (min_samples_split)**: If the number of samples in the current node is less than this value, the node will not be split further. This prevents nodes from splitting on very small subsets of data that may not generalize well.
- **Minimum Samples per Leaf (min_samples_leaf)**: After a split, each child node must contain at least min_samples_leaf samples. This parameter ensures that no leaf node is created with too few samples, which could lead to overfitting.

Building the Tree Recursively

The tree-building process works recursively as follows:

- **Step 1**: Start at the root node with the entire dataset.
- **Step 2**: Find the best feature and threshold that maximizes the information gain.
- **Step 3**: Split the data into two subsets based on the best threshold.
- **Step 4**: Create child nodes for each subset and recursively repeat the process for each child.
- **Step 5**: Stop the recursion if a stopping criterion is met. When stopping, assign the most common class label in the node to become the prediction for that leaf.

For example, imagine you have a dataset with features such as age, weight, and height, and you are predicting obesity levels. The decision tree starts by considering each feature and testing different values to see which split gives the highest information gain. Suppose splitting the data based on "weight <= 80 kg" yields the highest information gain. In that case, the dataset is split into two groups: individuals with weight less than or equal to 80 kg and those with weight greater than 80 kg. The process then repeats for each subset until a stopping condition is reached.

Results and Evaluation

After training the model, we evaluated its performance on the test set. Key metrics for evaluation included accuracy, precision, recall, and the f1-score and the entropy at every depth.

Confusion Matrix Visualization

To better understand the model's performance on each class, we used a **confusion matrix**. The **confusion matrix** shows how well the model predicted

each obesity class and highlights areas of misclassification, giving insights into possible improvements.

The **classification report** provided detailed metrics for each class:

- **Precision**: The proportion of positive identifications that were actually correct.
- **Recall**: The proportion of actual positives that were correctly identified.
- **F1-Score**: A harmonic mean of precision and recall, providing an overall effectiveness score.

The report indicated that most classes had high precision and recall scores, reflecting the model's effectiveness.

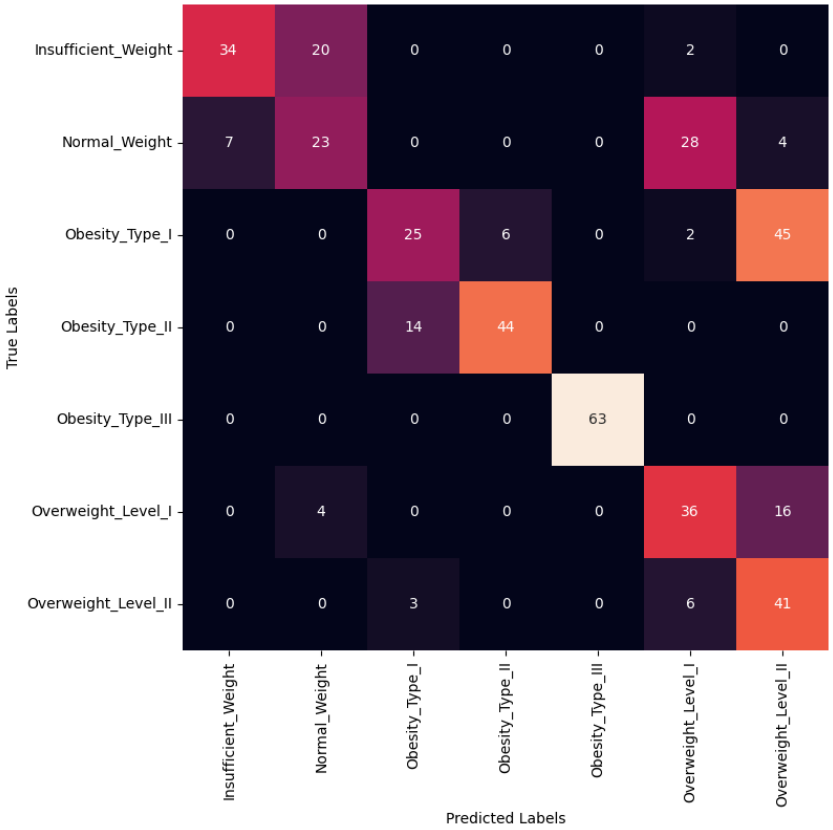
Test Cases for the Decision Tree Model

Case 1: Model with Low max_depth

- max_depth = 3
- min_samples_split = 5
- min_samples_leaf = 2

This case is designed to limit the model's complexity, which might make it less prone to overfitting, but could also cause it to miss important details.

Model Accuracy: 0.63%

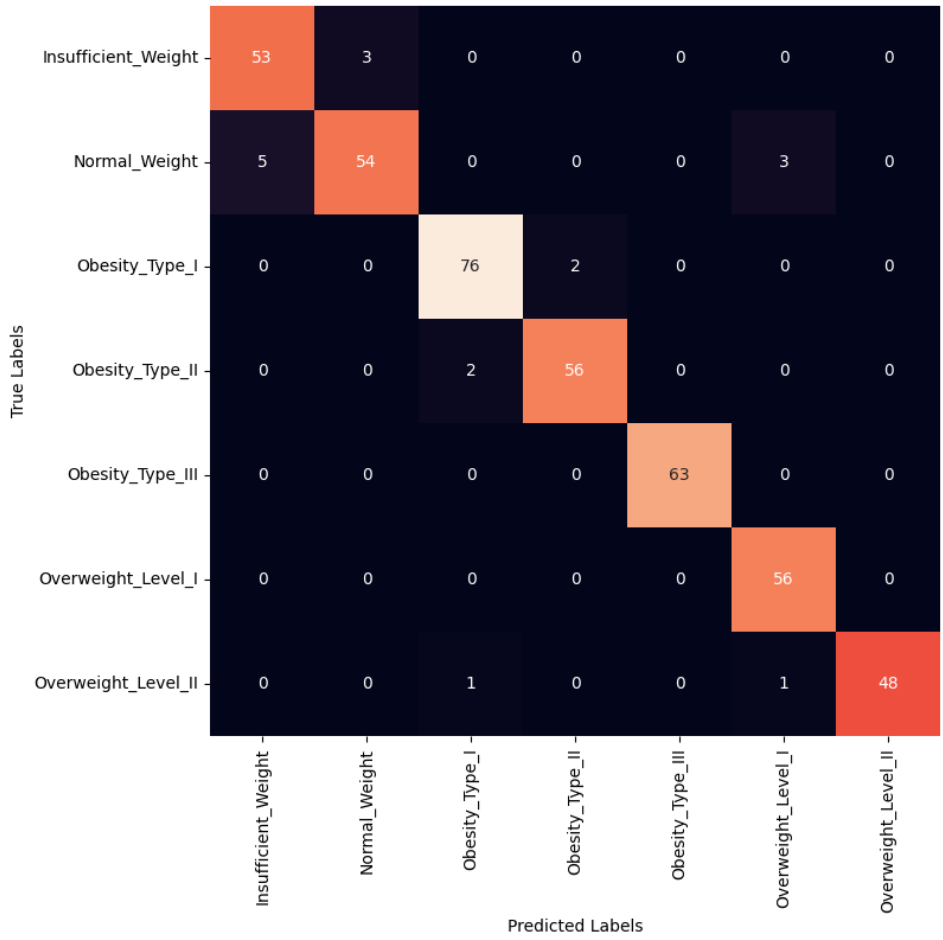


Case 2: Model with High max_depth and Adjusted Minimum Samples

- max_depth = 10
- min_samples_split = 4
- min_samples_leaf = 3

This case allows the model to grow deeper and have a higher capacity to fit complex data. This approach could help classify more complex classes better, but it might have a higher risk of overfitting.

Model Accuracy: 0.96%

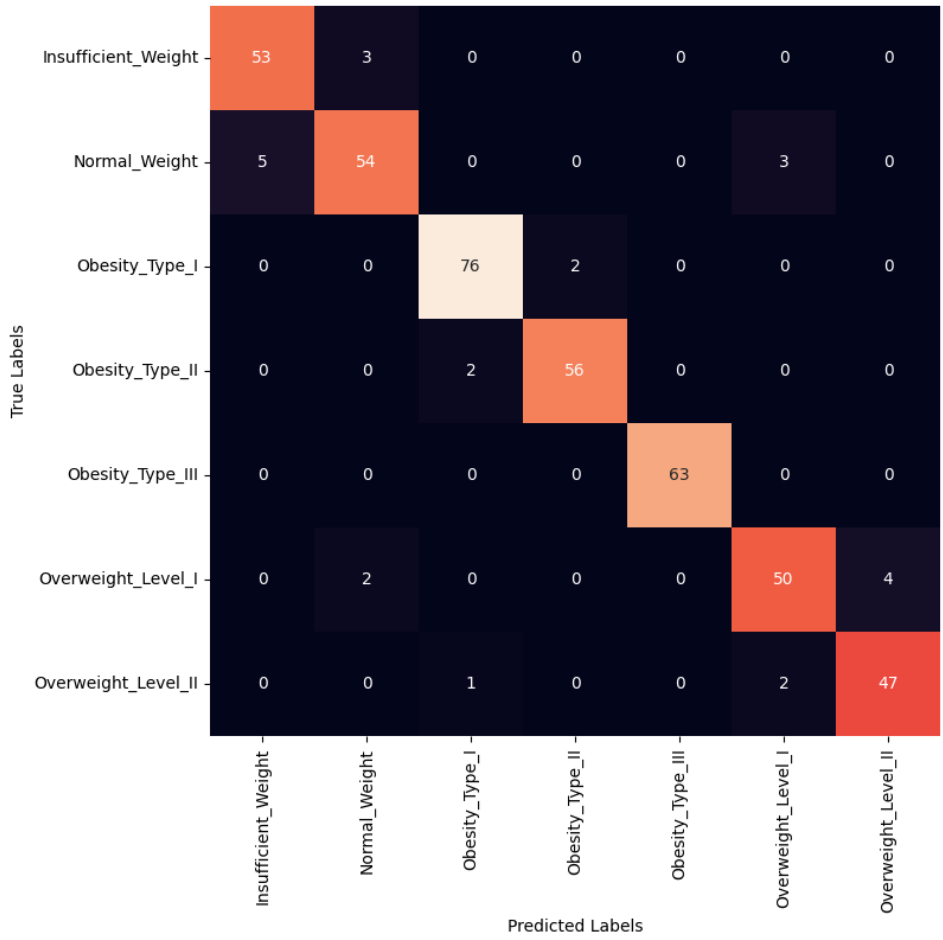


Case 3: Reducing Minimum Samples for Leaves

- max_depth = 8
- min_samples_split = 3
- min_samples_leaf = 1

In this case, min_samples_leaf is set to a low value, allowing leaves to be created with just one sample. This can lead to a very fine fit to the training data and might be beneficial if you want to capture specific patterns, but the risk of overfitting is higher.

Model Accuracy: 0.94%

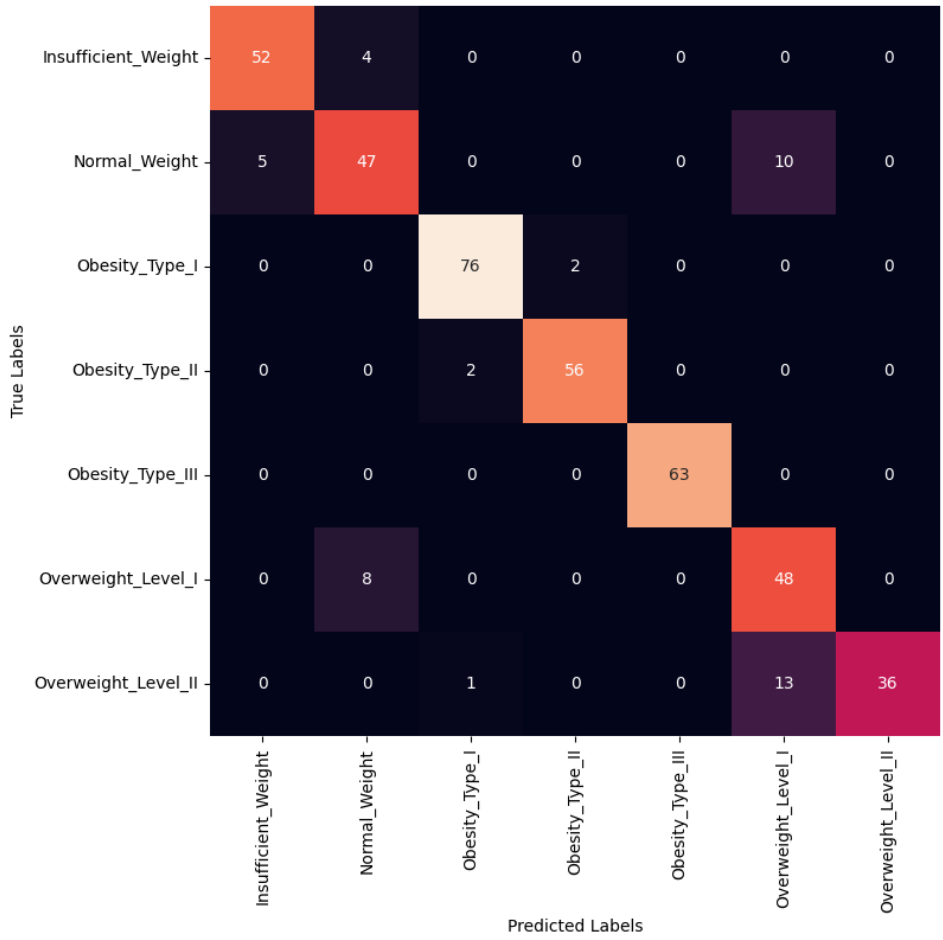


Case 4: Balanced Model

- max_depth = 6
- min_samples_split = 7
- min_samples_leaf = 4

This model has constraints on both depth and minimum number of samples, which helps prevent excessive growth and avoids branches with sparse data. This helps with generalization and prevents the model from overfitting to the specific training data.

Model Accuracy: 0.89%

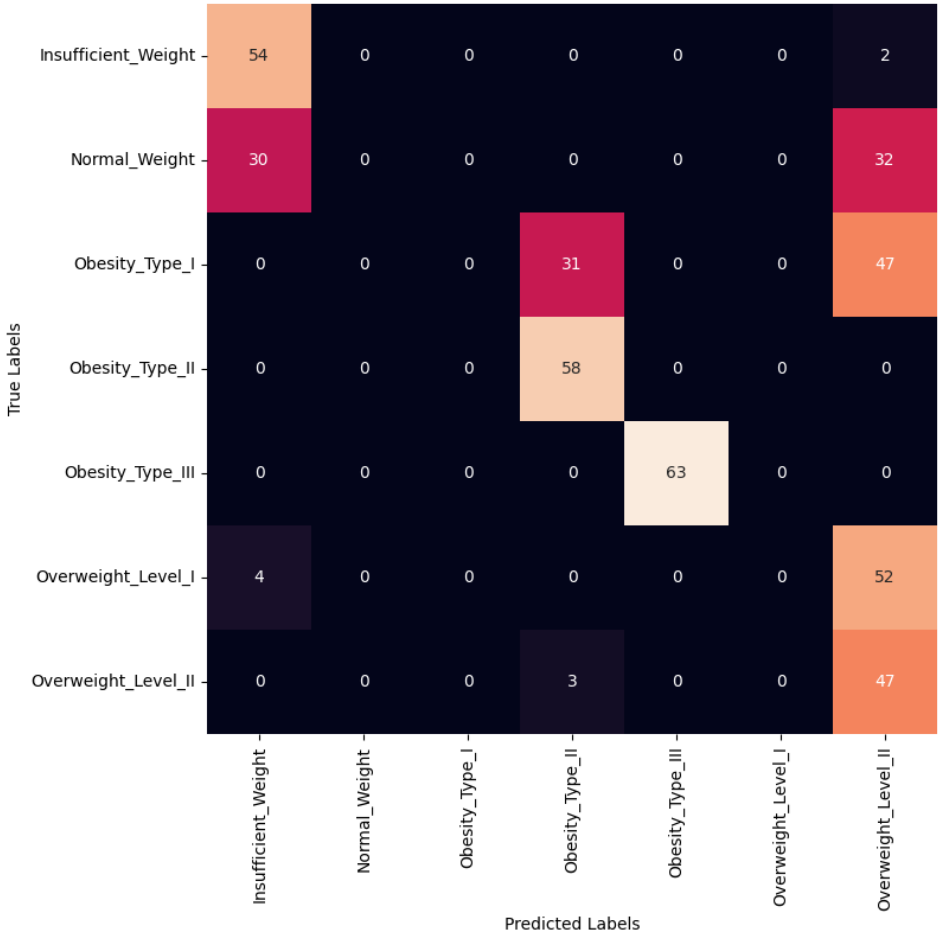


Case 5: Very Simple Model

- max_depth = 2
- min_samples_split = 10
- min_samples_leaf = 5

This is a highly constrained model, useful for seeing how a model behaves with high generalization capacity but low adjustment to data details. It would be interesting to analyze whether the model can still correctly identify the more general classes.

Model Accuracy: 0.52%



Conclusion

The implemented decision tree classifier demonstrated a strong performance in classifying obesity levels, achieving an accuracy of 95%. The model's depth and hyperparameter settings allowed it to effectively split data into homogenous groups, although some challenges in differentiating between certain classes were evident. This indicates that while the model successfully learned to generalize the main patterns in the dataset, there are still overlapping features among some obesity levels that lead to misclassifications.

References

Estimation of Obesity Levels Based On Eating Habits and Physical Condition [Dataset]. (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5H31Z.