Obesity Level Classification using Feedforward Neural Networks

Yiming Wang

November 10th, 2024

1 Introduction

In recent years, obesity has become one of the most widespread health concerns worldwide. According to the World Health Organization, about 1 in 8 people globally were estimated to be living with obesity in 2022. While obesity itself may not be immediately life-threatening, it is often linked to serious health issues like diabetes, cardiovascular diseases, and even certain cancers, making it essential to understand the factors contributing to its prevalence.

This project aims to analyze obesity levels based on eating habits and physical conditions in individuals by building a feedforward neural network model capable of classifying individuals based on their obesity risk. By examining a dataset covering individuals from Latin America, I focus on identifying meaningful patterns and developing a model that can accurately predict obesity levels based on factors such as age, diet, and physical activity.

2 Data Summary

This project uses a dataset published in 2019 by Fabio Mendoza Palechor and Alexis De la Hoz Manotas to study obesity levels among individuals in Mexico, Peru, and Colombia. The dataset includes 2,111 records with 17 attributes, which categorize individuals into seven levels of obesity, from Insufficient Weight to Obesity Type III.

About 23% of the data was collected directly from users, while the remaining 77% was generated using Weka and the SMOTE filter to maintain balance. The dataset includes key factors such as age, height, weight, diet, and activity habits, providing a detailed basis for examining obesity risks in these populations.

Variable	Type	Description	Responses
Gender	Categorical	Individual's Gender	- Female - Male
Age	Continuous	Individual's Age	Numeric Value
Height	Continuous	Individual's Height	Numeric value in meters
Weight	Continuous	Individual's Weight	Numeric value in kilograms
family_history_with_overweight	Binary	Has a family member suffered or suffers from overweight?	- Yes - No
FAVC	Binary	Do you eat high caloric food frequently?	- Yes - No
FCVC	Integer	Do you usually eat vegetables in your meals?	- Never - Sometimes - Always
NCP	Continuous	How many main meals do you have daily?	- Between 1 y 2 - Three - More than three
CAEC	Categorical	Do you eat any food between meals?	- No - Sometimes - Frequently - Always
SMOKE	Binary	Do you smoke?	- Yes - No
CH2O	Continuous	How much water do you drink daily?	- Less than a liter - Between 1 and 2 L - More than 2 L
SCC	Binary	Do you monitor the calories you eat daily?	- Yes - No
FAF	Continuous	How often do you have physical activity?	- I do not have - 1 or 2 days - 2 or 4 days - 4 or 5 days
TUE	Integer	How much time do you use technological devices such as cell phone, videogames, television, computer and others?	- 0-2 hours - 3-5 hours - More than 5 hours
CALC	Categorical	How often do you drink alcohol?	- I do not drink - Sometimes - Frequently - Always
MTRANS	Categorical	Which transportation do you usually use?	- Automobile - Motorbike - Bike - Public Transportation - Walking
NObeyesdad (Target)	Categorical	Obesity level	- Insufficient Weight - Normal Weight - Overweight Level I - Overweight Level II - Obesity Type I - Obesity Type II - Obesity Type III

Table 1: Table Summary of Variables

3 Data Preprocessing

The first step in preprocessing was to import and review the dataset's structure. The target variable, NObeyesdad (Obesity Level), was assigned to y, with all other columns forming the feature matrix X.

To prepare for model training and evaluation, the data into three sets: 60% for training, 20% for validation, and 20% for testing. This split allows tuning on the validation set while keeping the test set untouched for final performance assessment.

FCVC (Frequency of Vegetable Consumption) and TUE (Time Using Technology) were stored in float datatype, even though they represent whole numbers. These two features were converted to integers for later processing

Certain features, like family_history_with_overweight, FAVC, SMOKE, and SCC, were originally labeled as "yes" and "no." These values were converted to binary format (1 for "yes" and 0 for "no") to simplify model processing.

The target variable, NObeyesdad, was label-encoded to transform the obesity levels into numerical labels compatible with the neural network.

The following transformations were applied to the features:

- **Continuous Features:** Variables such as Age, Height, Weight, NCP, CH2O, and FAF were standardized to have a mean of 0 and a standard deviation of 1. This step was taken to normalize input ranges and improve model performance.
- **Binary Features:** Binary variables (family_history_with_overweight, FAVC, SMOKE, and SCC) had already been converted to binary format, so no further transformation was required.
- **Integer Features:** One-hot encoding was applied to FCVC and TUE to prevent any ordinal relationships.
- **Categorical Features:** Categorical features such as Gender, CAEC, CALC, and MTRANS were one-hot encoded, assigning each category a unique binary column for compatibility with the neural network.

A preprocessing pipeline was built using Column Transformer to ensure consistent transformations across the training, validation, and test sets. This pipeline was fitted on the training set and then applied to the validation and test sets. After transformation, the feature set shapes were confirmed as follows: Train shape: $(1266, 31)$, Validation shape: $(422, 31)$, and Test shape: $(423, 31)$.

These preprocessing steps ensure the data is clean, standardized, and ready for training the neural network to predict obesity levels.

4 Model Architecture

The Feedforward Neural Network used in this project was constructed using the Sequential API from TensorFlow's Keras library. The model is structured with four layers: three fully connected (dense) layers, followed by an output layer designed for multi-class classification. Each layer is configured as follows:

- **Input Layer:** The input layer receives 31 features.
- **Hidden Layers:**
  - **First Dense Layer:** This layer has 64 neurons and uses the Scaled Exponential Linear Unit (SELU) activation function. The weights are initialized using He Normalization, suitable for SELU to maintain a stable distribution of activations. A regularization term $\lambda = 0.001$ is applied to the kernel weights using L_2 regularization to prevent over fitting by penalizing large weights. We can represent the output of this layer as:

$$z^{(1)} = \text{selu}(\mathbf{W}^{(1)} \cdot \mathbf{X} + b^{(1)})$$

    where $\mathbf{W}^{(1)}$ is the weight matrix for the first layer, $\mathbf{X}$ is the input vector, $b^{(1)}$ is the bias vector, and $selu(x) = \lambda x$ if $x > 0$; otherwise, $\alpha(e^x - 1)$ where $\lambda \approx 1.0507$ and $\alpha \approx 1.67326$.
  - **Dropout Layer:** This layer randomly sets 50% of the layer's neurons to zero during training. This is done to prevent over fitting by forcing the network to learn patterns over iterations.
  - **Second Dense Layer:** This layer has 32 neurons and is configured the same was as the first dense layer. The output for this layer can be written as:

$$z^{(2)} = \text{selu}(\mathbf{W}^{(2)} \cdot z^{(1)} + b^{(2)})$$

  - **Third Dense Layer:** This layer has 16 neurons and is configured the same way as the second dense layer. The output for this layer can be written as:

$$z^{(3)} = \text{selu}(\mathbf{W}^{(3)} \cdot z^{(2)} + b^{(3)})$$

- **Output Layer:** This output layer is designed for multi-class classification with 7 neurons representing each obesity group. We apply the soft max activation function to normalize the output into a probability distribution, with each neurons output representing the probability of being a specific group.

The probability can be written as:

$$\hat{y}_j = \frac{e^{z_j}}{\sum_{k=1}^{7} e^{z_k}}$$

where $\hat{y}_j$ is the probability of the j-th class.

The model is then complied using the Nadam optimizer, an adaptive learning rate optimization algorithm that combines the benefits of RMSprop and momentum, with a learning rate of $\eta = 0.001$. The loss function "sparse categorical cross entropy" is used for multi-class classification with integer-labeled targets. Finally, accuracy was used to evaluate the model's performance during training and validation.

The model was trained for 200 epochs with a batch size of 32. Validation was used to monitor the model's generalization performance on unseen data during training.

The model leveraged SELU activation, HE Normal initialization, and regularization in attempt to stabilize training, reduce over fitting, and enhance classification performance.

5 Hyper parameter Tuning

The hyperparameter tuning was conducted using the Keras Tuner library, allowing exploration of a range of hyperparameters and testing different combinations to identify those yielding the best validation accuracy.

A custom function, named "skeleton_model," was defined to establish the model's architecture and test various combinations of hyperparameters. Specifically, the following hyperparameters were tuned:

- **Number of Hidden Layers:** Values tested ranged from 0 to 10.
- **Number of Neurons in Each Layer:** Values tested ranged from 16 to 256.
- **Learning Rate:** Values tested spanned a logarithmic scale from 1×10^{-4} to 1×10^{-2}.
- **Optimizer:** Four optimizers were tested: Adam, RMSProp, Nadam, and Adamax.

The skeleton model begins with a base Feedforward Neural Network model with no dropout layers or specific hyperparameter configurations. When the Random Search hyperparameter tuner from Keras is run, a new model is compiled for each trial with specified hyperparameter choices, effectively testing various combinations of network configurations. The function then evaluates each model's accuracy and loss, identifying the optimal set of hyperparameters for improved performance. This systematic tuning approach enhances model performance while managing issues of overfitting, underfitting, and computational cost.

After running the skeleton model and the Random Search method, the final tuned model suggested the following hyperparameters:

- **Number of Hidden Layers:** 9
- **Number of Neurons in each layer:** 124
- **Learning Rate:** 0.0005509513888645584
- **Optimizer:** Adamax

6 Results

The Training and Validation Loss, as well as the Training and Validation Accuracy, were plotted for both the untuned and tuned models to allow for a comparison of performance in each instance.

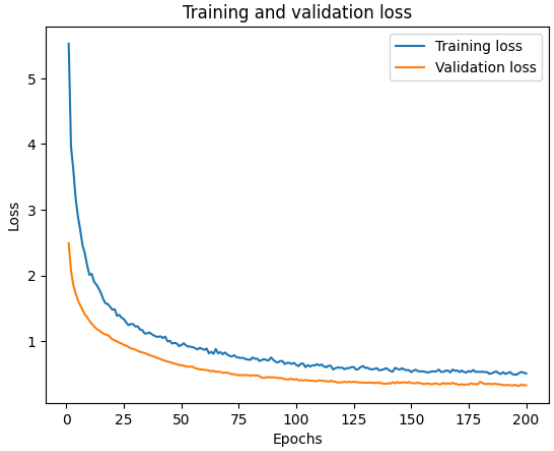

Figure 1: **Initial Model:** Training and Validation Loss

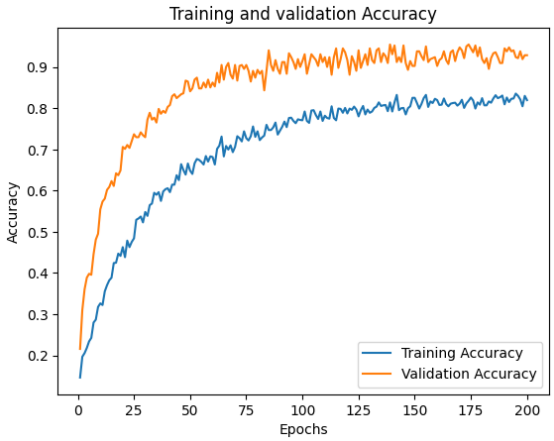

Figure 2: **Initial Model:** Training and Validation Accuracy

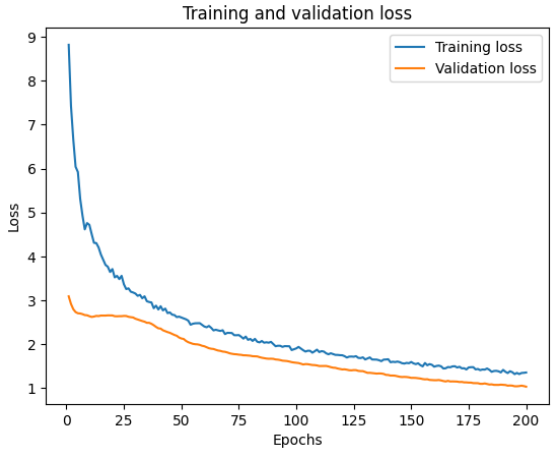

Figure 3: **Tuned Model:** Training and Validation Loss

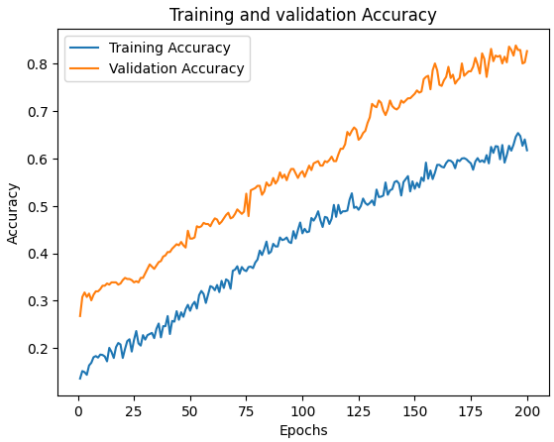

Figure 4: **Tuned Model:** Training and Validation Accuracy

Figures 1 and 2 show the results of the training and validation loss, as well as training and validation accuracy, for the initial model introduced in Section 4 with no hyperparameter tuning. Figures 3 and 4 display the results for the tuned model, which uses the "best" hyperparameters suggested in Section 5. The untuned model shows evidence of high bias, indicated by a large gap between training and validation loss and accuracy. In contrast, the tuned model reduces bias by optimizing hyperparameters, resulting in improved accuracy, reduced loss, and better alignment between training and validation performance.

The initial untuned model reaches a steep increase in accuracy more quickly (within 50 epochs) compared to the tuned model, which takes over 150 epochs to achieve around 75% accuracy. The validation loss for the untuned model consistently decreases over epochs, while the validation loss of the tuned model shows a slight increase around 25 epochs, indicating potential overfitting. Overall, the original model's use of dropout layers contributes to better generalization on unseen data.

7 Interpretation

Section 6 indicates that the original model achieves better test accuracy and test loss compared to the tuned model. This result highlights that while hyperparameter tuning can enhance Feedforward Neural Network performance, it does not always guarantee an optimal or near-optimal solution.

The training and validation results show that both loss and accuracy curves exhibit steady convergence over iterations. Decreasing loss values reflect strong learning and data-fitting capabilities, while increasing accuracy values suggest effectiveness in handling new, unseen data. Overall, the model demonstrates minimal issues with underfitting or overfitting and performs well in predicting future data. Thus, it can be reasonably concluded that the model effectively classifies an individual's obesity level based on their attributes.

References

[1] UCI Machine Learning Repository, *Estimation of Obesity Levels Based On Eating Habits and Physical Condition*, 2019. DOI: `https://doi.org/10.24432/C5H31Z`.

[2] Dr. Hojin Moon , *Lecture Notes on Neural Networks*, STAT 479: Applied Neural Network and Deep Learning, California State University, Long Beach, 2024.