Stat 479 Midterm Project

Cory Suzuki, Meha Patel, Brian Luu, Yiming Wang

November 2024

1 Introduction

Pondering the meaning of life and existence itself is nothing less than an arduous task. As a question that has eluded many great thinkers, scholars, and philosophers, many different schools of thought have arisen over the past centuries of human history, all trying to uncover this absolute truth. Without diving too deeply into philosophy, we may borrow an interpretation from the perspective of ethics and morals: the idea of living a long and fulfilling life.

Although simple-sounding, we begin to realize the difficulties of achieving this goal when considering from the perspective of global public health–how individuals' longevity may be challenged by circumstantial variables; variables describing an individual's physical characteristics and daily habits are extremely important in understanding overall health.

All around the world, obesity is a prevalent disease (WHO estimates in 2022 predicted that 1 in 8 people worldwide were living with obesity). Although obesity itself is not immediately deadly, the associated excessive fat deposits within the body often lead to increased risk of more detrimental conditions (e.g. diabetes, cardiovascular diseases, cancers, etc.)

For the scope of this paper, we study individuals' health by examining obesity levels as a result of different variables and aim to create a working neural network model to classify individuals accordingly.

2 Data Summary

We will utilize a prepared dataset from University of California, Irvine's Machine Learning Repository details obesity levels among individuals from Mexico, Peru, and Colombia, based on their dietary habits and physical condition. It is important to note that 23% of the data was collected from users and 77% from predictions (Weka Tool and SMOTE filter). This dataset includes 17 variables (16 features and 1 target) and 2111 observations. There are no missing values or any necessary cleaning to be done with the dataset, however the dataset did need to be preprocessed prior to model selection and validation. We provide a brief description of each of our feature variables below:

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 dataset was first imported and previewed to understand its structure and contents. The columns FCVC (Frequency of Consumption and Vegetables) and TUE (Time using technology devices) were initially in floating-point format. Since these variables represent whole-number values, they were converted to integers to reflect their true format.

We define y to be our target variable NObeyesdad (Obesity level) and separate it from the feature set. Therefore, the feature matrix X contains all the predictors while y contains only the target variable

The dataset was then split into training, validation, and testing sets to support model training and evaluation. The data was divided into the following: 60% training set, 20% validation set, 20% test set. This split allows us to tune our parameters for our model on the validation set, and finalize out model's performance assessment on the test set.

This dataset includes several binary categorical features (family_history_with_overweight, FAVC, SMOKE, and SCC) originally represented using values "yes/no". We convert these variables to a binary format (1 for "yes" and 0 for "no") across the training, validation, and test sets, to allow for easier model processing and enhanced computational efficiency.

Our target variable NObeyesdad was label encoded to transform the categorical obesity levels into numerical labels suitable for a Feed forward Neural Network model. This encoding was applied across the training, validation and test target variables.

The input features were transformed in the following ways:

- **Continuous Features:** Numerical continuous variables (Age, Height, Weight, NCP, CH2O, FAF) were standardized using StandardScaler to ensure a mean of 0 and a standard deviation of 1. This was done to improve the neural network performance by normalizing the input range.

- **Binary Features:** Since we already previously transformed binary variables (family_history_with_overweight, FAVC, SMOKE, and SCC) into 0's and 1's. We do not need to perform any further transformations.

- **Integer Features:** Although FCVF and TUE were already integer-based, one-hot encoding was applied to to prevent any ordinal relationship.

- **Categorical Features:** All categorical features (Gender, CAEC, CALC, and MTRANS) were one-hot encoded to expand each category into a unique binary value for model compatibility.

The preprocessing pipeline was defined using Column Transformer, allowing for a consistent and well-organized transformation of variables across training, validation, and test sets. The pipeline was fitted on the training set, and was then applied to the validation and test sets to ensure consistency in the feature transformation process. The transformed features sets were then checked to confirm the correct shapes after encoding, scaling, and splitting which resulted in the following: Train shape: $(1266, 31)$, Validation shape: $(422, 31)$, Test shape: $(423, 31)$.

The preprocessing helps ensure that the data is clean, standardized, transformed, and ready to be used to train and evaluate our Feedforward Neural Network model on obesity level prediction.

4 Model Architecture

The Feedforward Neural Network for this project was built using the Sequential API from TensorFlow's Keras library. The model consists of four layers, with the first three being fully connected (dense) layers, and the final layer being an output layer specifically for multi-class classification. The breakdown of each layer is 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.

This 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

We hyper parameter tune our model using the Keras Tuner library. This allows us to explore a range of hyper parameters and test different combinations to see which ones yield the best validation accuracy.

We defined a custom function: "skeleton_model", which defined the model's architecture and test difference combinations of hyper parameters. We specifically tried tuning the following hyper parameters:

- **Number of Hidden Layers:** Tested values between 0 and 10.

- **Number of Neurons in Each Layer:** Tested values between 16 and 256.

- **Learning Rate:** Tested values on a logarithmic scale from 1×10^{-4} to 1×10^{-2}.

- **Optimizer:** 4 different optimizers were tested: Adam, RMSProp, Nadam, and Adamax.

The skeleton model starts with a base Feedforward Neural Network model with no special dropout layers or hyper parameter specifications. Upon running the Random Search hyper parameter tuner from keras, we are able to compile a new model each trial with the choices of specified hyper parameters, effectively testing each combination of Feed forward Neural Network model parameters. The function then compares each model's accuracy and loss and determine the model with the "best" hyper parameters to use. This systematic hyperparameter tuning approach allows us to enhance model performance while controlling for over fitting, under fitting, and computational cost.

After running the Skeleton model and Random Search method, the suggested hyper parameters of the final tuned model are:

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

6 Results

We plot the Training and Validation Loss as well as the Training and Validation Accuracy for both the untuned and tuned model to compare its performance in both instances.

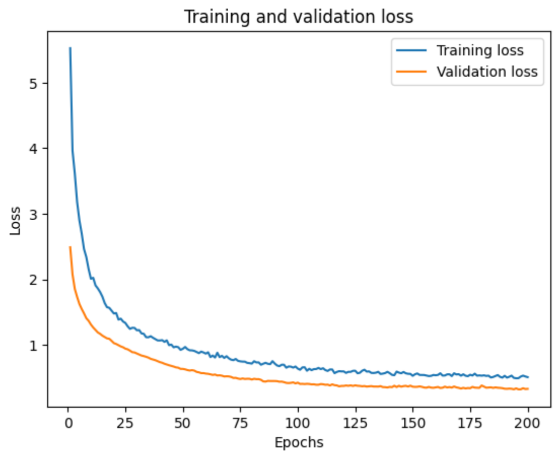

Figure 1: **Initial Model:** Training and Validation Loss

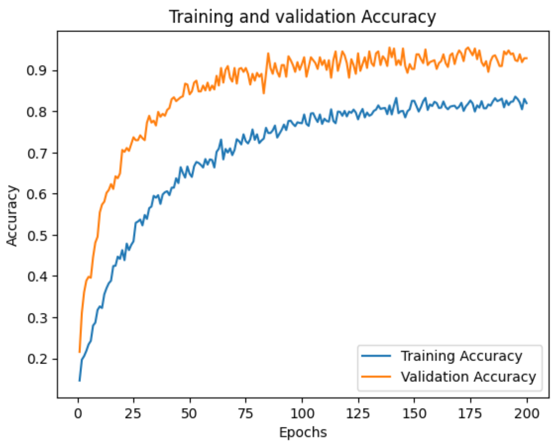

Figure 2: **Initial Model:** Training and Validation Accuracy

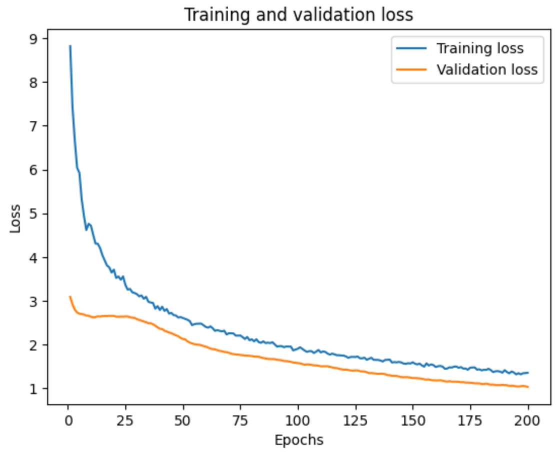

Figure 3: **Tuned Model:** Training and Validation Loss

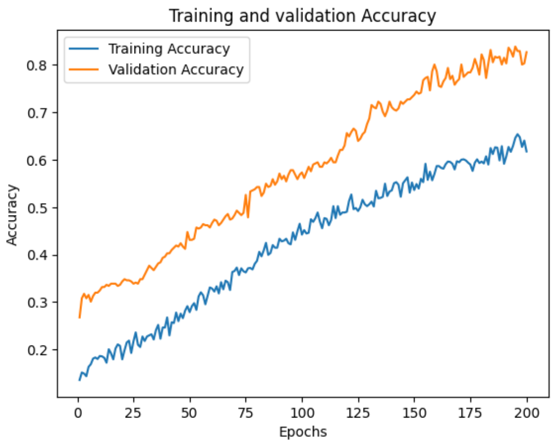

Figure 4: **Tuned Model:** Training and Validation Accuracy

Figures 1 and 2, show results of the training and validation loss and training and validation accuracy for the Initial model introduced in Section 4 with no hyper parameter tuning. Figures 3 and 4, show results of the training and validation loss and training and validation accuracy for the tuned model using the "best" hyper parameters suggested in Section 5. We notice the the untuned model displays evidence of high bias, indicated by the large gap between training and validation loss and accuracy. The tuned model effectively reduces the bias by optimizing hyper parameters, leading to improved accuracy and reduced loss, as well as better alignment between training and validation performance. We also notice that the initial, untuned model has a steep increase in accuracy rather quickly (within 50 epochs) compared to the tuned model, which takes over 150 epochs to reach around 75% accuracy. The validation loss for the initial model also exhibits a consistent decrease over epochs, whereas the validation loss of the tuned model has a slight increase around

25 epochs, indicating that it may be overfitting. Overall, the original model's use of dropout layers allows for a better generalization on unseen data.

7 Interpretation

From Section 6, we notice that our original model yields a better test accuracy and test loss compared to the tuned model, showcasing how hyperparameter tuning can yield better results for Feedforward Neural Networks, but will not guarantee an optimal or near optimal solution.

We examine our training and validation results and note that both loss and accuracy curves exhibit steady convergence over the iterations. Decreasing loss values indicate strong learning and data fitting abilities; similarly, increasing accuracy values signify suitability in handling new, future data. Overall, we conclude that our model has minimal issues with underfitting and overfitting and performs well for future predictions. Therefore, it is well within reason to say that our model is effective in classifying in new individual's obesity level status given 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.