Predicting Obesity Level using Neural Networks

Eashan Monga

McMaster University

DATASCI 3ML3

Dr. James Lambert

April 20, 2023

Table of Contents

Index of Figures

Index of Tables

Introduction

Motivation

One of the most prevalent issues facing the world today is the obesity epidemic. In 2016, 13% of the world's adults were obese, leading to over 4.7 million deaths globally in 2017 due to related risk factors. In addition, the obesity rate is expected to climb to 17.5% by 2030, which will cause about 1 billion people to have a significantly increased risk of chronic illness and/or death. Therefore, it stands to reason that finding the cause of obesity should be prioritized to prevent further disease and excessive deaths. Currently, obesity is caused by a variety of factors, including genetics, diet, physical activity, and prior health. However, the degree of contribution from each factor is still unknown, and therefore requires analyses to determine which factor is most closely correlated with obesity. This would allow for more focused treatment that prioritizes factors that contribute more to weight gain, and improve the ability of those afflicted with obesity to reduce their likelihood of other illnesses. On the other hand, conducting an analysis of the factors that contribute to obesity also creates better preventative treatment methods and may slow the rate of adults becoming obese or overweight. Therefore, conducting statistical analyses using a dataset of obesity risk factors and obesity diagnoses will help to identify the most important risk factors in obesity, and may help to improve the health of millions of people across the globe.

Objective

The objective of this project is to create a machine learning model capable of using common obesity risk factors to predict the obesity category, ranging from underweight to obese. The obesity risk factors will include dietary patterns, physical activity, lifestyles, and physical characteristics, like height and weight. After the analysis is completed, the model should be examined to determine which factors are more prevalent at higher

categories of obesity, and are better predictors of obesity.

Analysis of Problem

This problem involves input variables that consist of obesity risk factors, and output labels that consist of obesity categories. Most of these input variables consist of multiple numeric options that describe different levels within each variable. For example, smoking is registered as 0 for non-smoking and 1 for smoking, whereas daily water intake is divided between less than 1L, between 1-2 L, and greater than 2L. For variables such as smoking, one-hot encoding will be employed to categorize the inputs, whereas water intake will be coded continuously from 0 - 2. These variables would be used to find the feature vector to train the model, and would require an objective function that describes them in terms of multiple classes. Therefore, the goal of the analysis would be to divide the input vectors into the classifications as follows:

No.	Obesity Category
0	Insufficient Weight
1	Normal Weight
2	Obesity Level I
3	Obesity Level II
4	Obesity Level III
5	Overweight Level I
6	Overweight Level II

Table 1. *Overview of Obesity Categories in Order (Used in Confusion Matrices)*

These would be the labels of the function, and would be one-hot encoded in the order shown.

Finally, the analysis will consider which factors, organized into groups by lifestyle, diet, and physical activity, are more influential in determining the obesity label. In addition, since this problem will analyzed with the use of multi-class classification, the loss function will be categorical cross-entropy. This would be done to ensure that a mathematical optimization that minimizes the loss function would find parameters that

accurately and precisely predict the label of a given input for a classification problem.

Methodology

The model for obesity factors will be made using linear multi-class classification using neural networks. This model will be trained with a subset of the dataset under each obesity category to ensure model adequacy. Afterwards, it will be tested to see if it can correctly predict any obesity category given an input of the risk factors using a testing dataset. Finally, the model adequacy will be tested using confusion matrices and compared between various optimizer methods.

One of the serious limitations of this model may be the complex nature of obesity, and the factors that are not considered as inputs, such as genetics. In addition, the region is localized to South America, which may not represent the global population since climatic and geographical differences will lead to varying lifestyles and diets. Both of these factors could lead to the model being unable to accurately predict obesity labels or determine the obesity factors that affect everyone globally. In addition, other models may be better suited to describe the data if a linear relationship is not found, or is unable to successfully predict the obesity category. Thus, the highly complex nature of obesity and the limited dataset may lead to fitting problems requiring different model implementations. Finally, another problem that could occur would be the model predicting a close but incorrect obesity label. This may occur because the difference between Obese I, II, and III differs based on BMI, which is determined with height and density, both of which are somewhat unpredictable factors in physical composition. Overall, the model's real life implications may lead to complications with model adequacy, and may require testing and fitting to ensure a reliable fit.

Preliminary Results

Overall, this model accurately predicts obesity in the training dataset over 90% of the time using the Adam optimizer with one hidden layer. In addition, the confusion

matrix demonstrates that the model is capable of predicting the obesity level within a certain range. For example, if a person was labelled as normal weight, they would only be misclassified within a range of 1 step up or down, such as insufficient or overweight I. This implies that even when the model is not perfectly accurate, it is still capable of assessing the general obesity category of an individual. Furthermore, this result can also imply that these people may shift towards another obesity category over time since the inputs are based on their lifestyle. Therefore, it could be used to predict the lifestyle factors that eventually lead to obesity, which would help guide the population on what lifestyle factors they should choose to avoid obesity.

Methods

Dataset

In order to construct the obesity model, the data must first be processed and cleaned in order to ensure that each feature is coded correctly. First, the dataset is imported using Pandas to ensure that the dataset can be viewed and modified as necessary. Then, the dataset must be altered to accommodate continuous and categorical variables. Any variables that are deemed to be categorical are encoded using one-hot encoding, which forms vectors based upon the number of different options. For instance, if a variable has three different possibilities, then three vectors of length three will be created to classify these variables using 1's and 0's. In particular, each vector will only have one entry that corresponds to 1, whereas the rest of the entries will be 0. Therefore, each possible value for a categorical feature has a unique vector that can be used to create the appropriate weights for the model. This procedure will be done on any input and output variable that is categorical, as shown in **Table 8**.

In addition, the dataset will be split into three sections: training, testing, and validation, where 80% of the dataset will be used to train the model, 10% will be used to validate and 10% will be used to test the model. To ensure that the model testing is not biased towards a certain category, the dataset will be randomly sorted to ensure that an approximately uniform sample is taken for each section.

Neural Networks

The model will be constructed as a multi-class classification neural network using TensorFlow and Keras. As shown in **Figure 11** below, neural networks consist of an input layer, multiple hidden layers, and an output layer. In particular, the input layer is the layer that is constructed using the input data, which has 31 inputs as a result of the one-hot encoding in this model. Then, the input data is fed through the hidden layers that have a

varying number of inputs in order to better categorize and build the model. The final output layer corresponds to the desired categorical output, which is made up of the seven different categories of obesity in this particular model. The dataset is fed into the model as the input layer, and is run through the entire model over successive iterations, referred to as epochs. Each time the dataset goes through the neural network, the loss function is computed to compared the predicted and actual output. The closer the loss function is to 0, the closer the predicted and actual outputs, which signals a more accurate model. In order to minimize the loss function, the weights corresponding to each node are adjusted on each epoch using the optimizers. The specific discussion of each of these components is covered below:

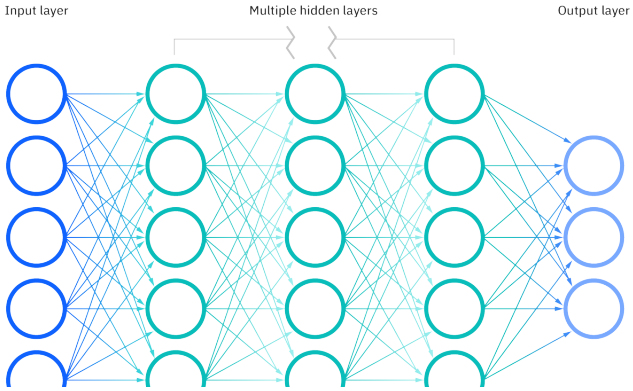

Figure 1. *Example of a Neural Network*

Layers and Activation Functions

As discussed earlier, a neural network consists of layers that help to categorize data using a variety of parameters. Each circle in **Figure 11** is referred to as a node, where the inputs to that node are weighted in order to create an output using the activation function. At each node, the activation function is used to determine if the node is "on" or "off", which essentially means if the combination of the inputs meets the threshold to activate the node similar to a biological neuron. This output will be used in the successive nodes in the next layer, which will once again use another activation function and weights to create an

output. This process will continue until the output layer, which is a vector that describes the output from a given input.

In terms of the activation functions, there are a variety of choices built into TensorFlow that have differences with regards to complexity and output. For this model, the SoftMax and Rectified Linear Unit activation functions were highlighted in particular, and are summarized below:

SoftMax:

$$P(y = i|\mathbf{x}) = \frac{e^{\mathbf{w}_i \mathbf{x}^T}}{\sum_{j=1}^{n} e^{\mathbf{w}_j \mathbf{x}^T}} \tag{1}$$

Where

$P(y = i|\mathbf{x})$: value of the SoftMax function for the ith class

$\mathbf{w}_i$: matrix of weights corresponding to class i

$\mathbf{x}$: input vector for function

n : number of classes

The SoftMax function can be described as a function that returns the weighted probability of each output using a specific input vector, deriving it's name from the smooth approximation of the argmax function. This activation function is particularly useful for classification problems, since it takes a vector and reports the probability of each component based upon their magnitude on a 0-1 scale. In turn, this will produce outputs that can form a vector where the components are similar to a one-hot encoded vector, where the largest input has the closest probability to 1, and every other input should be closer to 0. In addition, the SoftMax function has a weight matrix which will be adjusted to minimize the loss function in the output layer. In addition, the SoftMax function is smooth and differentiable everywhere, which helps with optimization methods. This will be used in the final output layer to determine the probability of each obesity category given

the input vectors since it is smooth and produces outputs from 0-1.

Rectified Linear Unit (ReLU):

$$f(\mathbf{x}) = max(0, \mathbf{w}\mathbf{x}^T) \tag{2}$$

Where

$f(\mathbf{x})$: value of ReLU for input vector $\mathbf{x}$

$\mathbf{w}_i$: matrix of weights

$\mathbf{x}$: input vector for function

The ReLU function compares the linear combination of the weights and input to 0, and returns the maximum between the two. Thus, the node only reports a non-zero output if the linear combination is large enough. This function also is computationally simple, and does not suffer from the vanishing gradient problem like other activation functions. Within this model, the ReLU function is used for the hidden layers to reduce complexity and ensure that nodes containing more important information are prioritized.

Loss Functions

For this model, the most applicable loss function is the categorical cross-entropy, due to the large number of classes and the requirement for a function that is used for classification. This equation is minimized to determine the weights that should be used in the final output layer, using inputs from the previous layer. It is very similar to the SoftMax function, and thus will measure the distance between the SoftMax output and the actual label vector. This loss function is summarized below:

$$g(\mathbf{W}) = -\frac{1}{P}\sum_{p=1}^{P} log\left(\frac{e^{\mathbf{w}_{yp}\mathbf{x}_p^T}}{\sum_{c=0}^{C-1} e^{\mathbf{w}_c\mathbf{x}_p^T}}\right) \tag{3}$$

Where

$g(\mathbf{W})$: categorical cross-entropy loss output

P : total number of training data points

C : total number of classes

$\mathbf{W}$: matrix of weights

$\mathbf{x}_p^T$: input vector for pth training data point

Optimizers

For this model, all of the optimizers are based upon gradient descent algorithms that are used to find the weights that minimize the loss function. These optimizers will generally find the gradient of the function using back-propagation and use it to find the weight matrix such that the loss function becomes minimized. A list of the optimizers and a brief overview of their use are shown below:

- Adaptive Moment Estimation (Adam) Optimizer: Adam uses a combination of two other optimization methods, momentum gradient descent and root mean square propagation (RMSprop), in order to find the optimal weights. In particular, momentum gradient descent uses the exponentially weighted average of past gradients in order to find the minimum without slowing down due to zig-zagging. On the other hand, the RMSprop method normalizes the components of a direction by using the exponential average of the component-wise magnitudes of the previous gradient directions. Adam combines these two methods and calculates exponential averages for both the direction and the magnitude of the descent.

- Stochastic Gradient Descent (SGD): SGD uses a variant of gradient descent that relies on a randomly selected data points to find the potential direction.

- Adaptive Gradient Descent (Adagrad): Adagrad is a variant on stochastic

optimization that adapts the learning rate based on the features themselves. In particular, it will update the parameters more frequently for features that occur less often, and vice versa.

- Root Mean Square Propagation (RMSProp): As mentioned in Adam, RMSProp normalizes the components of a direction by using the exponential average of the component-wise magnitudes of the previous gradient directions. In other words, it keeps an average of the previous gradient directions in order to guide future movements.

Accuracy Metrics

For this model, the most relevant accuracy metrics are the confusion matrix and the testing accuracy. The confusion matrix is constructed by using the predicted and actual labels and comparing them to see how often the model predicted categories correctly and incorrectly. For this model, it will be a 7x7 matrix that shows the difference between the actual obesity label and the predicted one. The general version is shown below, where the rows are the actual labels and the columns are the predicted ones.

Labels	0	1	2	3	4	5	6
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
4	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
5	(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)
6	(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)

Table 2. *Confusion Matrix for Seven Class Classification Problem*

Each (i, j) entry corresponds to the number of outputs that were predicted to be i, but are actually j. In particular, the diagonal entries of this matrix are the correctly predicted labels, whereas every other entry is an incorrect prediction. This will be used to assess the model adequacy and determine if obesity can be predicted.

Another important accuracy metric is the testing accuracy and loss, which measures the loss and the rate of correct predictions on the testing dataset. This will be used to test whether or not the chosen model parameters can be extended past the training and validation dataset, which is important since high model accuracy on those datasets may not translate to a model with high applicability and usability. In addition, overfitting is an issue that inhibits the ability of the model to predict new labels from a testing dataset, and should be examined with this metric to test the model.

Results

In order to examine the differences between optimizers and varying parameters, multiple different models were constructed. These models are all tested using the same parameters whenever possible to ensure comparability. To begin, the results from each model are summarized below their respective headings:

Adam Optimizer

The results of the Adam optimizer are shown in the figures and tables below:

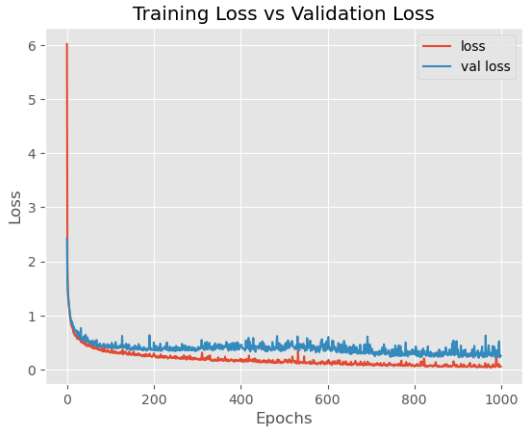

Figure 2. *Training vs Validation Loss for Adam*

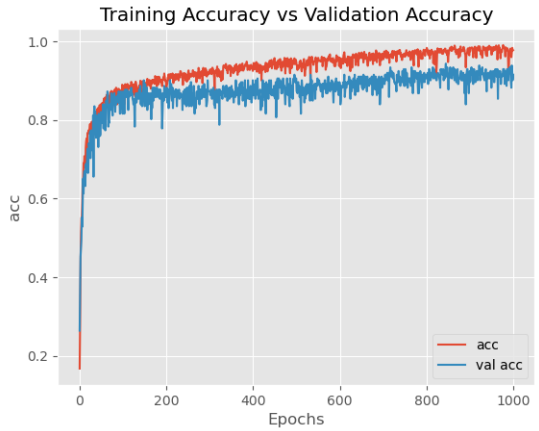

Figure 3. *Training vs Validation Accuracy for Adam*

25	0	0	0	0	0	0
1	26	0	0	0	1	1
0	0	34	0	0	0	0
0	0	0	28	2	0	0
0	0	0	0	29	0	0
0	0	1	0	0	29	0
0	0	2	0	0	4	28

Table 3. *Confusion Matrix for Adam*

For this optimizer, the testing loss was 0.3668 and the testing accuracy was 93.4%, which indicates that this model is very accurate. In addition, the confusion matrix shows that most of the categories are predicted correctly, with some deviation with the Overweight 1 category. Finally, the graphs show that using 1000 epochs does not lead to over fitting as the loss continually decreases over time. Overall, this is the best optimizer for this dataset, and will be explored further in the discussion.

SGD Optimizer with Same Parameters as Adam

The results of the SGD optimizer with the same parameters as Adam are shown in the figures and tables below:

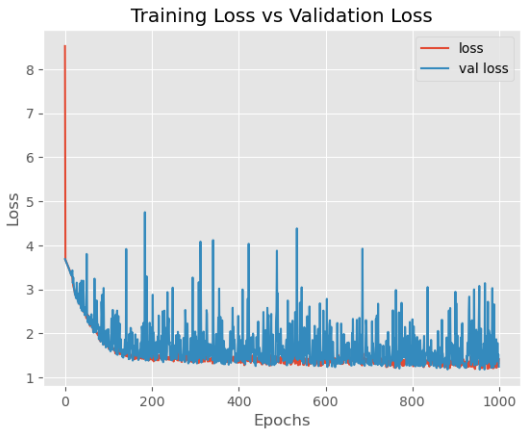

Figure 4. *Training vs Validation Loss for SGD*

For this optimizer, the testing loss was 1.28346 and the testing accuracy was 50.2%, which indicates that this model is not as accurate as Adam. In addition, the confusion

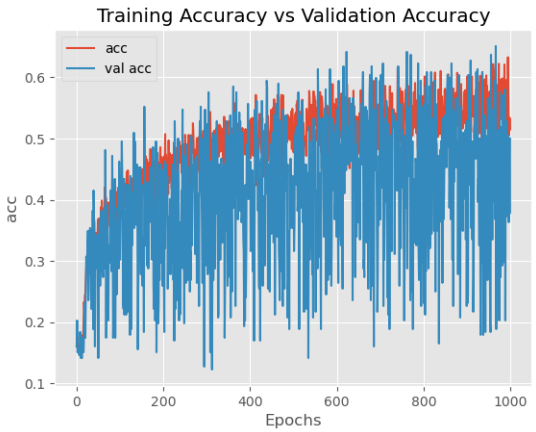

Figure 5. *Training vs Validation Accuracy for SGD*

25	0	0	0	0	0	0
17	10	0	0	0	2	0
0	0	18	0	5	3	8
0	0	19	7	3	1	0
0	0	0	0	29	0	0
0	13	0	0	0	15	2
0	9	9	0	0	14	2

Table 4. *Confusion Matrix for SGD*

matrix shows many of the categories not predicted correctly, especially towards the obese 1 and 2 categories. However, the insufficient and obese 3 categories are predicted the same as Adam, which may occur due to the nature of these two categories as extremes of the obesity range. Finally, the graphs indicate severe variance and range in the loss and accuracy for the training and validation dataset as expected from stochastic gradient descent. They do follow a trend of increasing over time; however, they are not as stable or accurate as Adam, and thus are not as useful for predicting obesity.

Adagrad Optimizer

The results of the Adagrad optimizer are shown in the figures and tables below:

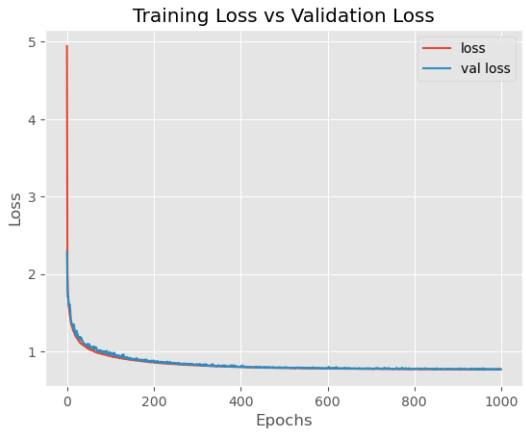

Figure 6. *Training vs Validation Loss for Adagrad*

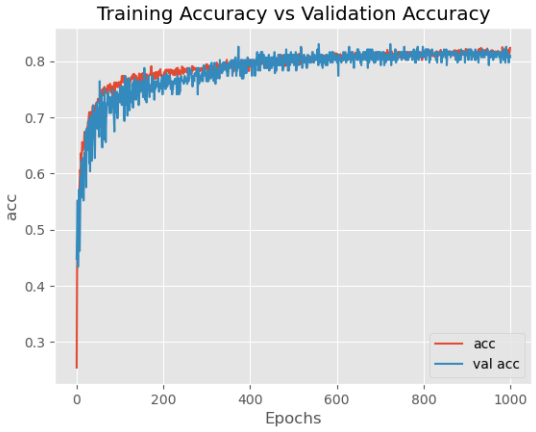

Figure 7. *Training vs Validation Accuracy for Adagrad*

25	0	0	0	0	0	0
9	16	0	0	0	4	0
0	0	33	0	0	0	1
0	0	1	29	0	0	0
0	0	0	0	29	0	0
0	6	1	0	0	19	4
0	0	5	0	0	11	18

Table 5. *Confusion Matrix for Adagrad*

For this optimizer, the testing loss was 0.82965 and the testing accuracy was 80.1%, which indicates that this model is decently accurate. In addition, the confusion matrix shows many of the categories are predicted correctly, except overweight I and II.

Interestingly, the insufficient and obese 3 categories are predicted the same as Adam and SGD, which may occur due to the nature of these two categories as extremes of the obesity range. Finally, the graphs have the lost variance and noise of all the optimizers, and demonstrate that there is no overfitting as of now.

RMSProp

The results of the RMSProp optimizer are shown in the figures and tables below:

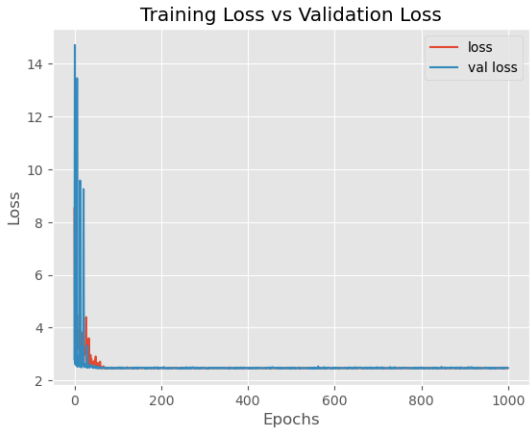

Figure 8. *Training vs Validation Loss for RMSProp*

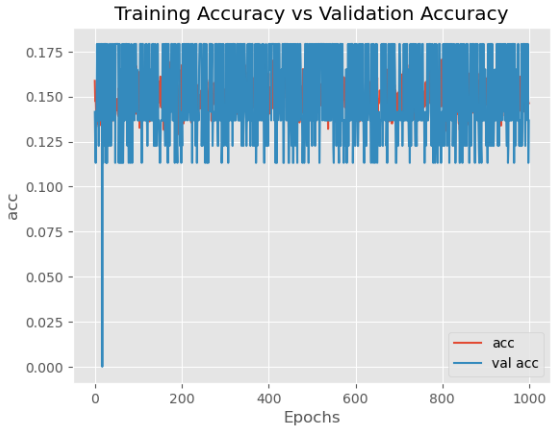

Figure 9. *Training vs Validation Accuracy for RMSProp*

For this optimizer, the testing loss was 2.4642 and the testing accuracy was 13.7%, which indicates that this model is no better than guessing. This is clearly visible in the

0	0	0	0	25	0	0
0	0	0	0	29	0	0
0	0	0	0	34	0	0
0	0	0	0	30	0	0
0	0	0	0	29	0	0
0	0	0	0	30	0	0
0	0	0	0	34	0	0

Table 6. *Confusion Matrix for RMSProp*

confusion matrix, which shows that the model incorrectly predicted every input to be obese I. This probably occurred due to a non-optimized set of parameters for RMSProp, and thus the optimizer became stuck in a non-optimal solution. This can also be seen the graphs of loss and accuracy, since it appears that the maximum accuracy and loss are attained quite quickly. Thus, RMSProp is not suitable for this dataset because it did not demonstrate the ability to reach a good solution for the weights. This most likely occurs due to the complex nature of the input dataset, which means that any method that isn't correctly configured to finding minima in higher dimensions will struggle to find a solution, like RMSProp.

SGD Optimizer with Multiple Hidden Layers

The results of the SGD optimizer with multiple hidden layers are shown in the figures and tables below:

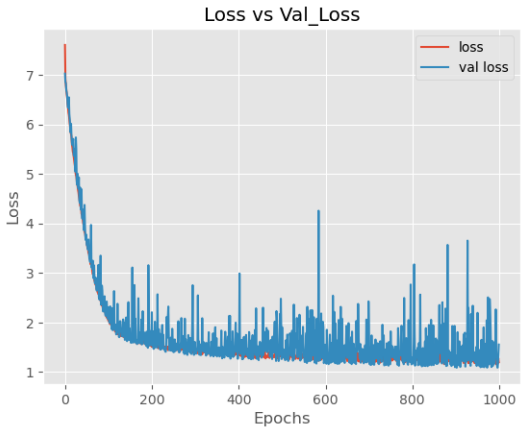

Figure 10. *Training vs Validation Loss for SGD with Hidden Layers*

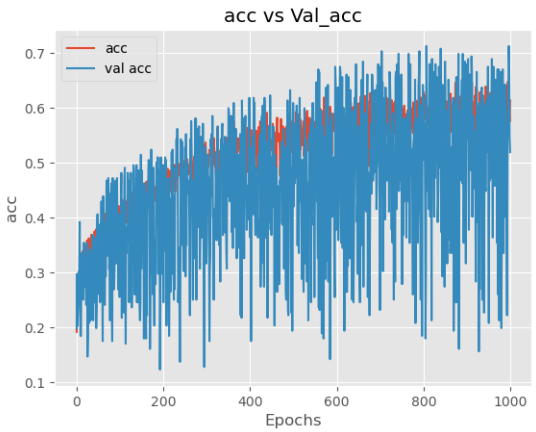

Figure 11. *Training vs Validation Accuracy for SGD with Hidden Layers*

25	0	0	0	0	0	0
12	14	1	0	0	2	0
0	0	13	0	16	3	2
0	0	4	0	26	0	0
0	0	0	0	29	0	0
0	8	7	0	0	13	2
0	2	13	0	1	17	1

Table 7. *Confusion Matrix for SGD with Multiple Hidden Layers*

For this optimizer, the testing loss was 1.7195 and the testing accuracy was 45.0%, which indicates that this model is not better than SGD without any hidden layers. This is also shown in the confusion matrix, which appears to have more deviation in various categories than the other SGD implementation. This probably occurred due to the unnecessary complexity of hidden layers, which can sometimes help to improve the model if the dataset is better suited to it. However, since SGD without hidden layers did not perform well nor did it require more complexity, adding hidden layers only decreased the accuracy and worsened the model predictions. This can also be seen the graphs of loss and accuracy, which appear to have more noise and variance, but no evidence of overfitting. Thus, the added layers do not help to create the model as the SGD optimizer does not work well with the dataset, especially compared to Adam.

Conclusion

Summary of Findings

From the results in the previous section, it is clear that the best optimizer is Adam using one hidden layer. It had the lowest loss and the highest testing accuracy, and according to the confusion matrix, predicted most of the categories correctly. This is because Adam combines multiple gradient descent methods to prevent zig-zagging while also ensuring that each step taken considers past steps. Furthermore, this combination most likely allows Adam to ignore the various local minima that exist throughout the model plane, which is probably why it performed much better than any other method. In particular, it was able to find a very good minima for the loss function compared to the other optimization methods, many of which do not have the computational capability of Adam. However, with more experimentation of the parameters, layers, and learning rates, other optimizers could be improved to reach levels that are adequate, such as Adagrad, which was able to reach 80% accuracy.

In addition, nearly all of the confusions matrices shared the same 1st and 5th diagonal entry, corresponding to insufficient weight and obesity III respectively. This most likely occurs because these two categories are on the extreme ends of the scale, which means that it is most likely easier to optimize for these categories since they have defining characteristics that other ones will not have. This implies that there are vast differences between those that are insufficient in weight or obese III as compared to other individuals, which makes sense since they probably do not have a healthy balance of food and physical activity. Therefore, this implies that there are defining characteristics of unhealthiness which can be taken from the model and used to ensure that people do not reach the extreme ends of the obesity scale.

Finally, the loss graphs demonstrate that all of the models do not suffer from overfitting, which implies that the number of epochs may be increased in order to improve

the model. This also probably stems from the fact that the model complexity is too great for any algorithm to over fit to every input point within 1000 epochs. In addition, these graphs also show varying degrees of noise between the models that originates due to the stochastic nature of the optimizers as well as the large dimension of the input space. This also inhibits the potential benefits for hidden layers, since it was shown that they do not improve the model adequacy. In this case, they only serve to slow down the computation, leading to less suitable output as compared to one without hidden layers.

Implications

Since the best model was able to predict obesity levels with over 93%, it is clear that there are defining features and lifestyle characteristics for those that are obese. One of the major issues with health care is the variance that occurs within biology that may be less predictable due to the sheer complexity of humans themselves. However, it is known that food consumption habits, current weight, and exercise routines are some of the most important factors to diagnosing and preventing obesity. This model confirms that there are definite ways to quantify obesity and predict it based upon certain lifestyle factors. In addition, this also confirms that obesity is not an individualistic disease that affects people differently; people who live a lifestyle towards obesity will tend to be more obese. Furthermore, the confusion matrices show that extreme ends of the obesity scale are vastly different to those that are contained within. Thus, there are also defining characteristics of those that are not of a healthy weight, and thus should change their lifestyles to improve their health.

Another key implication of this model is contained within the confusion matrix for Adam. In particular, whenever the model is inaccurate, it is only within 1 step of the true label. For example, if a person is labelled as overweight I, they may be classified as normal weight or overweight II. This implies that there is a degree of subjectivity within the model with regards to biology. However, this can also imply that these people may be living

lifestyles that may push them towards another obesity category. As a result, the incorrect predictions can be studied to see whether or not someone tends towards the predicted category over time, which would further strengthen the model as an accurate predictor of obesity. Thus, even the inaccuracy of the model helps to categorize and predict the obesity category of individuals.

Overall, the model is able to accurately predict those that have obesity, and potentially help prevent those that having lifestyle factors that may push them in that direction. Therefore, studying the model further and building will help to create an even better visualization of the factors that impact obesity and what should be done.

Next Steps

For the next steps, this model can be analyzed further by removing certain key variables, such as weight, to see whether or not weight plays a very significant role in predicting obesity. In addition, other variables or factors can be removed to see if the model becomes more or less accurate, which would help to elicit the most important factors for obesity. Finally, more data should be collected and tested, especially in varying climates and regions of the world. This would help to refine the model's capability with regards to regional differences and increase the applicability of the model.

Overall, the applications of neural networks in health care is limitless. Cancer, viral infections, and even genetic disorders could benefit from machine learning analyses. Not only can these models help to find factors that lead to disease, but they can potentially be used to find cures themselves. Therefore, this obesity model has demonstrated that neural networks and AI should be implemented more often into health care in the fight to improve the quality of life and longevity of humanity.

Appendix

Summary of Features

Features	Possible Values and Units	Continuous/Categorical
Gender	Female, Male	Categorical
Age	Numerical value	Continuous
Height	Numerical value in meters	Continuous
Weight	Numerical value in kilograms	Continuous
Family member that suffered or suffers from overweight	Yes, No	Categorical
Frequency of consumption of high caloric food (FAVC)	Yes, No	Categorical
Frequency of consumption of vegetables in meals (FCVC)	Never, Sometimes, Always	Categorical
Frequency of daily main meals (NCP)	Between 1 to 2, 3, More than 3	Continuous
Frequency of food between meals (CAEC)	No, Sometimes, Frequently, Always	Categorical
Smoking	Yes, No	Categorical
Daily water consumption (CH2O)	Less than 1 L, Between 1 and 2 L, More than 2 L	Continuous
Daily calorie consumption monitoring (SCC)	Yes, No	Categorical
Frequency of physical activity (FAF)	None, 1 to 2 days, 2 to 4 days, 4 to 5 days	Continuous
Time using technological devices such as cell phone, videogames, television, computers, etc (TUE)	0–2 hours, 3–5 hours, More than 5 hours	Continuous
Alcohol consumption frequency (CALC)	None, Sometimes, Frequently, Always	Categorical
Method of transportation (MTRANS)	Automobile, Motorbike, Bike, Public Transportation, Walking	Categorical
Obesity Category - Output Variable (NObeyesdad)	Insufficient Weight, Normal Weight, Obesity Level I, Obesity Level II, Obesity Level III, Overweight Level I, Overweight Level II	Categorical

Table 8. *Summary of Features and Variable Types for Obesity Model*

References

De-La-Hoz-Correa, E., Mendoza-Palechor, F. E., De-La-Hoz-Manotas, A., Morales-Ortega, R. C., & Beatriz Adriana, S. H. (2019). Obesity level estimation software based on decision trees. Journal of Computer Science, 15(1), 67–77. https://doi.org/10.3844jcssp.2019.67.77

Estimation of obesity levels based on eating habits and physical condition. UC Irvine Machine Learning Repository. (n.d.). Retrieved February 20, 2023, from https://archive-beta.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

Palechor, F. M.,& Manotas, A. de. (2019). Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico. Data in Brief, 25, 104344. https://doi.org/10.1016/j.dib.2019.104344

Ritchie, H., & Roser, M. (2017, August 11). Obesity. Our World in Data. Retrieved February 20, 2023, from https://ourworldindata.org/obesity

Watt, J., Borhani, R., & Katsaggelos, A. K. (2020). Machine learning refined: Foundations, algorithms, and applications. Cambridge University Press.

World Obesity Atlas 2022. World Obesity Federation. (n.d.). Retrieved February 20, 2023, from https://www.worldobesity.org/resources/resource-library/world-obesity-atlas-2022