CMSC 210: Assignment #3

Neural Networks

Neural Network Design

In general, the proponents took a trial-and-error approach to model the best Neural Network in this assignment. Similar to previous submissions, the approach is to develop a base model with default parameters and one with hyperparameter-tuned values. Details of the NN design are shown in Table 1.

Table 1. Neural Network Design

Consideration	Models						Description
	(A1) Default*	(A2) Best**	(B1) Default*	(B2) Best**	(C1) Default*	(C2) Best**	
Normalized or Scaled by	[0, 1]		[-1, 1]		[0, + ∞)		According to the AI Summer School (2017)[1], the NN model works best when scaled as such.
Input	(q7) Frequency of Telehealth Consultations, (q8) Time Spent on Health Applications (in hrs.), (q9) Trust in Telehealth (scale 1-5), and (q10) Digital Health Literacy Score (score 0-100) (*# of neurons: 4)*						The inputs are all four numeric attributes in the survey similar to the one used in the KNN model. The target value on the other hand describes the level of digital health adoption with three classes.
Output	(q1) Target Attribute: Level of Digital Health Adoption (Low, Moderate, and High) (*# of neurons: 3)*						
No. of Hidden Layers (neurons)	1 (10)	1 [1, 5, 10, 50, 100]	1 (10)	1 [1, 5, 10, 50, 100]	1 (10)	1 [1, 5, 10, 50, 100]	This assignment only required one hidden layer and 10 neurons (default). A combination of neurons was also explored.
Activation Function	Sigmoid	Sigmoid, Tanh, and ReLU	Tanh	Sigmoid, Tanh, and ReLU	ReLU	Sigmoid, Tanh, and ReLU	Various activation functions were used to fit and explore the models to test various scenarios.
Loss Function	Cross-entropy						This fxn was used due to being a classification prob.
Optimizer	Adaptive Moment Estimation	Lbfs, sgd, and adam	Adaptive Moment Estimation	Lbfs, sgd, and adam	Adaptive Moment Estimation	Lbfs, sgd, and adam	Adam is the default optimizer in Python. However, a combination of methods was also explored.
Hyper Parameters	Batch size: n_samples, Learning rate: 0.001, and Epoch: 200	Please see code below	Batch size: n_samples, Learning rate: 0.001, and Epoch: 200	Please see code below	Batch size: n_samples, Learning rate: 0.001, and Epoch: 200	Please see code below	The hyperparameters were based on the performance of the base models with default parameters, and these were extended ± to a certain extent --- this excludes the epoch (default= 200) due to high computing time.

Note: (*) Base model with default parameters. (**) Hyperparameter tuned model.

[1] Based on the Powerpoint presentation during the Neural Network Model lectures in the AI Summer School of 2017 conducted by the Philippine Department of Science and Technology.

```
# -------------- Step 3: Define hyperparameters for hyper parameter tuning -------------
param_grid = {
    'hidden_layer_sizes': [(1,), (5,), (10,), (50,), (100,)],
    'activation': ['logistic','relu', 'tanh'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate_init': [0.001, 0.01, 0.1],
    'max_iter': [200],
    'solver':['lbfgs', 'sgd', 'adam'],
    'batch_size': [50, 100]
}
```

Fig. 1. NN Parameter Grid for Hyperparameter tuning.

Neural Network Architecture

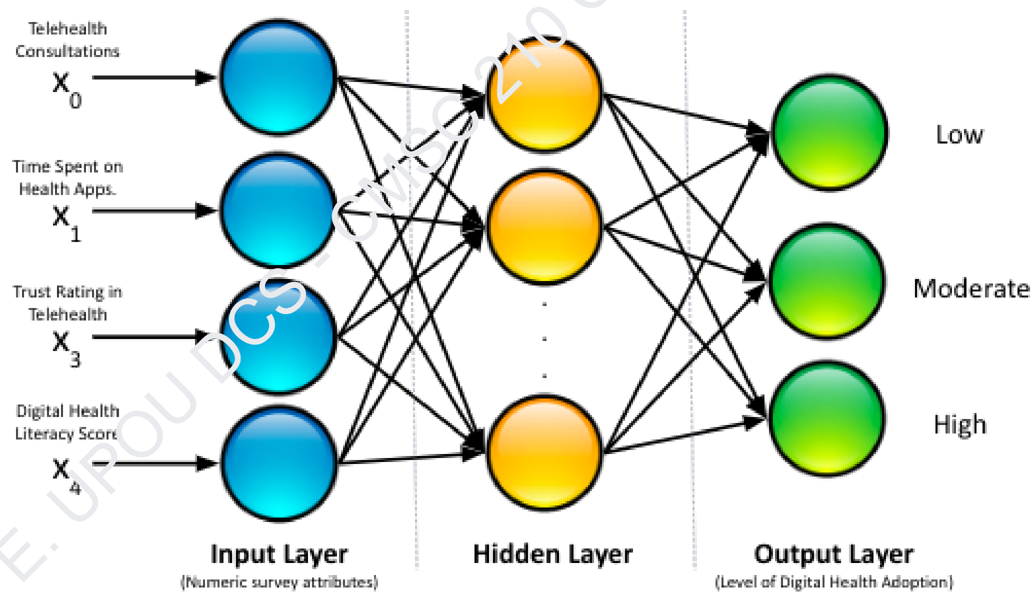


Fig. 2. Neural Network Architecture for the "Digital Health Adoption Survey" numeric data.

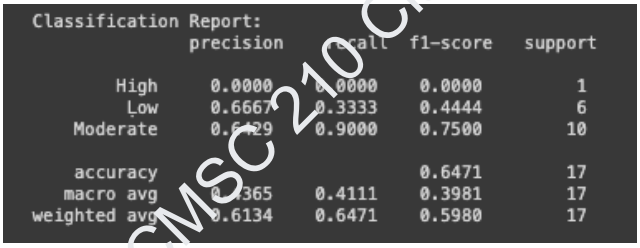
Neural Network Performance

There were six models developed and the performance for each is shown in Table 2. From this result, we can say that none of the models has performed exceptionally well. It is also observed that for the base models (default) their accuracy on the testing data set > training dataset which most likely suggests that there is an issue with the split percentage and/or imbalance (Only three for the "High" class) in the dataset. So, to compromise, the base models were not considered and the model to be used and discussed in the succeeding section of this paper will be the model with the highest testing accuracy among the hyperparameter tuned cases which is the "(C2) Best" model. This considers better generalization to unseen data and potential over-fitting due to varied accuracies.

Table 2. Model Performance

Metric	Neural Network Models					
	(A1) Default*	(A2) Best**	(B1) Default*	(B2) Best**	(C1) Default*	(C2) Best**
Accuracy on Training dataset	51.35%	86.49%	70.27%	62.16%	67.57%	**70.27%**
Accuracy on Testing dataset	58.82%	41.18%	52.94%	64.71%	70.59%	**64.71%**

Note: (*) Base model with default parameters. (**) Hyperparameter tuned model.



Classification Report:

	precision	recall	f1-score	support
High	0.0000	0.0000	0.0000	1
Low	0.6667	0.3333	0.4444	6
Moderate	0.6429	0.9000	0.7500	10
accuracy			0.6471	17
macro avg	0.4365	0.4111	0.3981	17
weighted avg	0.6134	0.6471	0.5980	17

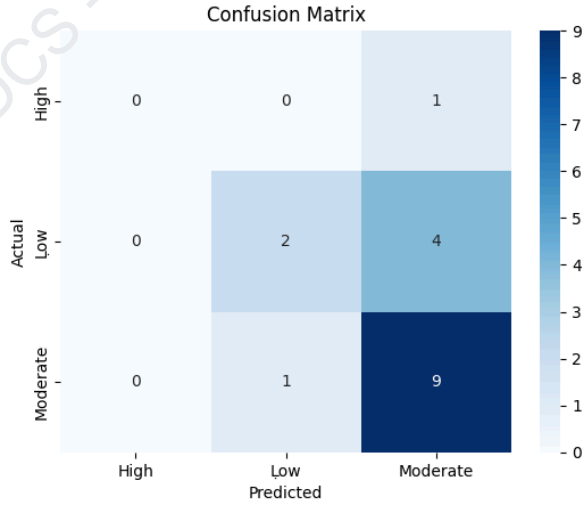
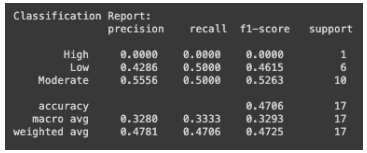
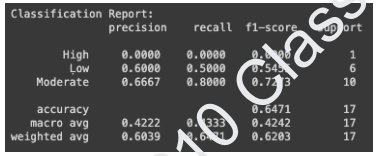
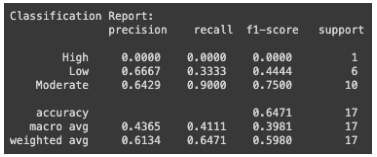
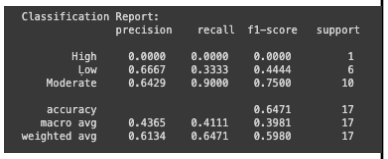


Figure 2. Neural Network Performance of model '(A2) Best'. (Top: Classification Report, Bottom: Confusion Matrix)

Performance Comparison

Table 3. Performance comparison of the three "best" classifiers.

Results		Hyper Param. Decision Tree (Best)	Hyper Param. KNN (Best - Scaled)	Hyper Param. Naïve-Bayes Classifier (Best)	Neural Network '(C2) Best'
Accuracy on Training		94.59%	**70.27%**	64.86%	**70.27%**
Accuracy on Test Set		47.06%	**64.71%**	64.71%	**64.71%**
Classification Report					
Standard Cross Validation	*Scores*	[0.55, 0.45, 0.55, 0.55, 0.4]	[0.82, 0.64, 0.64, 0.27, 0.7]	[0.73, 0.55, 0.55, 0.45, 0.4]	[0.82, 0.64, 0.36, 0.73, 0.5]
	Avg. Score %	50%	**61%**	53%	**61%**
Stratified Cross Validation	*Scores*	[0.36, 0.55, 0.55, 0.64, 0.2]	[0.82, 0.45, 0.73, 0.55, 0.7]	[0.82, 0.6, 0.64, 0.73, 0.5]	[0.82, 0.45, 0.64, 0.73, 0.7]
	Avg. Score %	46%	65%	61%	**67%**
Shuffle-split Cross Validation	*Scores*	[0.47, 0.41, 0.35, 0.65, 0.65]	[0.65, 0.59, 0.41, 0.59, 0.71]	[0.65, 0.47, 0.35, 0.65, 0.71]	[0.65, 0.65, 0.35, 0.59, 0.59]
	Avg. Score %	51%	**59%**	56%	56%

How does the Neural Network fare against the other methods?

The Neural Network demonstrates a comparable performance to the K-Nearest Neighbor and Naive Bayes models, with moderate accuracy levels on both the training and test datasets as shown in Table 3. While the Decision Tree model exhibits overfitting, the Neural Network maintains a better balance between training and test accuracies. Cross-validation results consistently support the Neural Network's reasonable generalization capabilities and performance is quite similar to the K-Nearest Neighbor model's performance. Its classification report also shows slightly better performance VS the Decision Tree; and similar level of performance with the Naive Bayes and K-Nearest Neighbor. Overall, the Neural Network does not significantly outperform other methods in comparison.

Is a Neural Network classifier as explainable as Decision Tree classifier? Why or why not?

NN models are less explainable compared to Decision Trees due to their complex, non-linear structures and lack of intuitive feature importance measures. The black-box nature of NN, influenced by intricate learned representations and non-linear activations, limits their ability to provide clear insights into how input features contribute to predictions.

Further Analysis

Do you think a Neural Network classifier with more hidden layers and hidden layer neurons will perform better (i.e., higher accuracy)? Why or why not?

This depends on the (classification) problem and the data we input into our model. So, the answer is not a direct yes or no. Adding more hidden layers and neurons might help the model learn intricate patterns and representations. However, this also suggests increasing the complexity of the NN model, which might lead to overfitting, increased computational complexity, and potential difficulties in training.

In literature, some studies have yet to mention that more hidden layers and neurons are better regardless of the problem. However, studies like Karsoliya (2012) mentioned that *"...taking a suitable number of hidden layers and the number of neurons in each hidden layer, better results in less training time can be obtained."* On the other hand, Uzair and Jamil (2020) mentioned that finding the correct number of hidden layers and neurons is still a difficult task. However, for large and complex problems, satisfactory accuracy results can be achieved. A different perspective by Panchal et al. (2011) mentioned ways of selecting the appropriate number of hidden layers and neurons. Nevertheless, one of the highlights in their paper is that they mentioned:

"For nearly all problems, one hidden layer is sufficient. Two hidden layers are required for modeling data with discontinuities such as a saw tooth wave pattern. Using two hidden layers rarely improves the model, and it may introduce a greater risk of converging to a local minima. There is no theoretical reason for using more than two hidden layers."

Studies also show that having an excellent configuration for the number of hidden layers and hidden layer neurons is more than just a one-size-fits-all solution. A study shows that having less than three hidden layers may reduce accuracy and minimal model complexity, stating poor network training. Having the right balance for the optimal hidden layer increases the computational problem, making the dominant accuracy. As highlighted by Jason Brownlee (2019) "*A single-layer neural network can only be used to represent linearly separable functions. This means very simple problems where, say, the two classes in a classification problem can be neatly separated by a line. If your problem is relatively simple, perhaps a single layer network would be sufficient.*" He added, "*A Multilayer Perceptron can be used to represent convex regions. This means that in effect, they can learn to draw shapes around examples in some high-dimensional space that can separate and classify them, overcoming the limitation of linear separability.*" Moreover, Karishma Gupta states, "*The number of hidden layers is highly dependent on the problem and the architecture of neural network.*" Concerning neurons, Sandhya Krishnan (2021) states, "*Most of the problems can be solved by using a single hidden layer with the number of neurons equal to the mean of the input and output layer. If less number of neurons is chosen it will lead to underfitting and high statistical bias. Whereas if we choose too many neurons it may lead to overfitting, high variance, and increases the time it takes to train the network.*" This highlights the importance of understanding the data and adopting the trial and error approach to identify the optimal order for hidden layers and neurons in the neural network. This is why it is essential to boost data-driven exploration to achieve the best result for your data.

Common to these studies, they have mentioned that deciding the number of hidden layers and neurons in each hidden layer is still confusing, and the number depends on the dataset and problem. Thus, more does not necessarily mean better accuracy, but it might lead to such, especially for complex problems. It should also be noted that increasing/decreasing the number of hidden layers and neurons is not positively/negatively correlated with the model's accuracy.

```
Accuracy on training set: 64.86%
Accuracy on test set: 64.71%
1. Mean Test Score: 0.6179, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (5,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
2. Mean Test Score: 0.5714, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (5, 5), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
3. Mean Test Score: 0.2964, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (5, 5, 5), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
4. Mean Test Score: 0.3786, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (5, 5, 5, 5), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
5. Mean Test Score: 0.6000, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (5, 5, 5, 5, 5), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
```

```
Accuracy on training set: 70.27%
Accuracy on test set: 64.71%
1. Mean Test Score: 0.5143, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (6,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
2. Mean Test Score: 0.6571, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (10,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
3. Mean Test Score: 0.4107, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (25,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
4. Mean Test Score: 0.5214, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (50,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
5. Mean Test Score: 0.517[illegible], Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (100,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
6. Mean Test Score: 0.5750, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (250,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
7. Mean Test Score: 0.[illegible]750, Hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'batch_size': 50, 'hidden_layer_sizes': (500,), 'learning_rate_init': 0.001, 'max_iter': 200, 'solver': 'sgd'}
```

Figure 3. Neural Network Performance of model '(A2) Best' with tuned hidden layers and neurons. (Top: Best tuned model train and set accuracies and model tuning results by increasing the number of hidden layers, Bottom: Best tuned model train and set accuracies and tuning results by increasing the number of hidden neurons)

In this exercise, we also attempted to show this phenomenon by tuning the best Neural Network model by increasing the number of hidden layers and neurons. The assumption is that we have the same best parameters except that the hidden layers (from one to five, with five neurons each) and neurons (from six to 500, with only one hidden layer) are explored. As shown in Figure 3, the results suggest that increasing the number of hidden layers and neurons can increase the accuracy score; however, determining the suitable number is still a conundrum. It is noticeable in the results that

increasing the number of layers or neurons does not necessarily show a positive linear relationship but fluctuates for each new case.

In conclusion, the optimal architecture depends on the complexity of the data and the specific task. It also often requires experimentation and careful tuning of parameters to strike a balance between model complexity and generalization to achieve better performance.

References

Brownlee, J. (2019, August 6). *How to Configure the Number of Layers and Nodes in a Neural Network*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/.

Gupta, K. *Hidden Layers*. https://iq.opengenus.org/hidden-layers/

Karsoliya, S. (2012) Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *International Journal of Engineering Trends and Technology, 3(6),* 714-717.

Krishnan, S. (2021, September 9). *How do determine the number of layers and neurons in the hidden layer?* Medium. https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3.

Uzair, M., and Jamil, N. (2020). Effects of Hidden layers on the Efficiency of Neural Networks. 2020 IEEE 23rd International Multitopic Conference (INMIC). Access last November 26, 2023 from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9318195.

Panchal, G., Ganatra, A., Kosta, Y.P., and Panchal, D. (2011). Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers. *International Journal of Computer Theory and Engineering, 3*, 332-337.