**Module 19 Challenge Report**

**Neural Network**

**Background**

Artificial Neural Networks or simply put Neural Networks is another technique of machine learning that recognize feature patterns in a set of input data, in the process provides a measured and quantified output. The Neural Network technique consist of collections of layered self-computing interconnected nodes. Output of one is input to another node.

**Objective**

The objective is to build a machine learning model capable of analyzing and predicting the likelihood of a company or companies being successful using the provided (charity\_data.csv) dataset. The essence of this is to use the information to support a clean and clear decision-making process with respect to which organization(s) are worth donating money to. Furthermore, we want to attempt to optimize and evaluate for better performance.

**Development Environment** Python Pandas TensorFlow Jupyter Notebook Data file (charity\_data.csv)

**Code Plan**

The following is recommended guide for the code development. Some of the steps may be combined to achieve the same result. Note that the amount of preprocessing required depends on the how clean/dirty the data is.

1.Import dependencies and libraries.

2.Import the input dataset.

3.Generate categorical variable list.

4.Create a OneHotEncoder instance.

5.Fit and transform the OneHotEncoder.

6.Add the encoded variable names to the DataFrame.

7.Merge one-hot encoded features and drop the originals.

8.Split the preprocessed data into features and target arrays.

9.Split the preprocessed data into training and testing dataset.

10.Create a StandardScaler instance.

11.Fit the StandardScaler.

12.Scale the data.

13.Define the model.

14.Add first and second hidden layers.

15.Add the output layer.

16.Check the structure of the model.

17. Compile

18. Train

19. Evaluate

20. Tweak parameters for optimization purposes and evaluate

**Summary**

Based on the size of the data and the input variables (features) I have 118 Neurons with 2 Hidden Layers. Hidden Layer1 = 6, Hidden Layer 2 = 4. Epoch set to 100.

The first attempt of the evaluation yielded approximately 73% accuracy with the loss at about 55%. Loss: 0.5583208229381906, Accuracy: 0.7334110736846924

**Steps taken to optimize the model and increase performance** **(in bold One parameter at a time)**:

1. Hidden Layer 1 = 6 **Hidden layer 2 = 3** Epoch set to 200

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_3 (Dense) (None, 6) 708

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 3) 21

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 1) 4

=================================================================

Total params: 733 Trainable params: 733 Non-trainable params: 0

Epoch 199/200

25724/25724 [==============================] - 1s 35us/sample - loss: 0.5434 - acc: 0.7325

Epoch 200/200

25724/25724 [==============================] - 1s 36us/sample - loss: 0.5432 - acc: 0.7326

8575/8575 - 3s - loss: 0.5547 - acc: 0.7334

Loss: 0.5546591354320071, Accuracy: 0.7334110736846924 Outcome = No improvement

1. Hidden Layer 1 = 8 Hidden layer 2 = 4 **Epoch set to 200**

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_6 (Dense) (None, 8) 944

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 4) 36

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 1) 5

=================================================================

Total params: 985 Trainable params: 985 Non-trainable params: 0

Epoch 98/100

25724/25724 [==============================] - 1s 35us/sample - loss: 0.5353 - acc: 0.7392

Epoch 99/100

25724/25724 [==============================] - 1s 35us/sample - loss: 0.5352 - acc: 0.7378

Epoch 100/100

25724/25724 [==============================] - 1s 37us/sample - loss: 0.5351 - acc: 0.7378

8575/8575 - 0s - loss: 0.5529 - acc: 0.7339

Loss: 0.5529235258324848, Accuracy: 0.7338775396347046 Outcome = No improvement

**(c)** Hidden Layer 1 = 8 Hidden layer 2 = 4 Epoch set to 100 **Activation = Tanh**

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_9 (Dense) (None, 8) 944

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_10 (Dense) (None, 4) 36

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_11 (Dense) (None, 1) 5

=================================================================

Total params: 985 Trainable params: 985 Non-trainable params: 0

Epoch 98/100

25724/25724 [==============================] - 1s 36us/sample - loss: 0.5331 - acc: 0.7405

Epoch 99/100

25724/25724 [==============================] - 1s 35us/sample - loss: 0.5329 - acc: 0.74090

Epoch 100/100

25724/25724 [==============================] - 1s 37us/sample - loss: 0.5329 - acc: 0.7416

8575/8575 - 0s - loss: 0.5456 - acc: 0.7306

Loss: 0.5456023949228292, Accuracy: 0.7306122183799744 Outcome = No improvement

Optimization attempts were made by changing the values of hidden layer 1, hidden layer 2, Epoch and Activation. These attempts are highlighted in a, b and c above.

There is a noticeable slight upward trend when the **EPOCH** was increased to **1000**. However, due to limited computational power the result is not conclusive. I intend to continue to pursue this further on Google Colab to see if higher EPOCH could make a difference.

Epoch 998/1000

804/804 [================] - 1s 1ms/step - loss: 0.5241 - accuracy: 0.7439

Epoch 999/1000

804/804 [================] - 1s 1ms/step - loss: 0.5247 - accuracy: 0.7433

Epoch 1000/1000

804/804 [================] - 1s 1ms/step - loss: 0.5243 - accuracy: 0.7431

268/268 - 0s - loss: 0.5475 - accuracy: 0.7331

Loss: 0.5475156307220459, Accuracy: 0.7330612540245056

* My other choice of machine learning model to implement for this type of classification will be the Support Vector Machine (SVM). The reason being the fact that SVM appears to be very robust with binary classification, as well as dealing with overfitting challenges. However, further investigation will be required to re-examine the input data to ascertain if the real problem is with binary classification or not.