**MYRNA A. DESEO**

# Assignment 21: Deep Learning

# Report on the Neural Network Model

## Overview

Alphabet Soup is a nonprofit foundation that funds new ventures for various organisations. Its business team has provided a dataset that captures the metadata about each organisation that had applied and Alphabet Soup would like to be able to predict which venture application will be successful to help Alphabet Soup in its assessment of applications.

This project made use of neural network to predict which applicants will be successful, by using cleaning the data, creating training and testing sets, and finally evaluate the models.

## Results

### *Data Processing*

The metadata was "cleaned' by removing the non-beneficial information or columns such as EIN and NAME columns.

The feature variables used for the model are: APPLICATION TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL CONSIDERATIONS and ASK_AMT. The target variable for the model is the column "IS_SUCCESSFUL," which is aimed to be predicted with high accuracy for this project.

The data was binned using the CLASSIFICATION value counts to reduce the number of unique categorical values. A list of CLASSIFICATION was created whereby any count that is less than 1,000 was grouped into a single "Other" category. The categorical data in the data frame was converted to numeric data, and the preprocessed data was split into a training and testing dataset. The training and testing features datasets were scaled by creating a "StandardScaler" instance and transformed using the "transform" function.

### *Compiling, Training, and Evaluating the Model*

A deep neural network model was created by assigning the number of input features and nodes for each layer using TensorFlow and Keras. For the initial analysis, 2 hidden layers with 80 neurons for Layer1 and 30 neurons for Layer2 were used with Relu activation function for both layers. The output node is a binary classification model and used a Sigmoid activation function as below:

```python
# Define the model - deep neural net
number_input_features = len(X_train[0])
hidden_nodes_layer1 =  80
hidden_nodes_layer2 = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="relu")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="relu"))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                3520
_________________________________________________________________
dense_1 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 31
=================================================================
Total params: 5,981
Trainable params: 5,981
Non-trainable params: 0
```

```python
# Compile the model

nn.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```python
# Train the model

fit_model = nn.fit(X_train_scaled, y_train, epochs=100)
```

```
Epoch 1/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5702 - accuracy: 0.7212
Epoch 2/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5553 - accuracy: 0.7290
Epoch 3/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5535 - accuracy: 0.7303
Epoch 4/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5512 - accuracy: 0.7317
Epoch 5/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5488 - accuracy: 0.7320
Epoch 6/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5488 - accuracy: 0.7331
Epoch 7/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5478 - accuracy: 0.7321
Epoch 8/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5480 - accuracy: 0.7341
Epoch 9/100
804/804 [==============================] - 1s 1ms/step - loss: 0.5470 - accuracy: 0.7340
```

The model was trained with 100 epochs and the model was evaluated using the test data, which resulted to 0.73 accuracy with 0.55 loss.

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5543 - accuracy: 0.7265
Loss: 0.5543338060379028, Accuracy: 0.7265306115150452
```

The model was optimized in an attempt to increase the accuracy:

*Optimisation 1*: A third hidden layer was added with 30 nodes.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                3520
_________________________________________________________________
dense_1 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_2 (Dense)              (None, 30)                930
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 31
=================================================================
Total params: 6,911
Trainable params: 6,911
Non-trainable params: 0
_________________________________________________________________
```

There was not much change in the accuracy and loss from the previous model:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5575 - accuracy: 0.7247
Loss: 0.5574899911880493, Accuracy: 0.7246647477149963
```

*Optimisation 2*: A fourth hidden layer was added with the nodes in each layer as shown below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                3520
_________________________________________________________________
dense_1 (Dense)              (None, 40)                3240
_________________________________________________________________
dense_2 (Dense)              (None, 30)                1230
_________________________________________________________________
dense_3 (Dense)              (None, 20)                620
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 21
=================================================================
Total params: 8,631
Trainable params: 8,631
Non-trainable params: 0
_________________________________________________________________
```

Adding a 4th layer did not make any difference in the accuracy that still showed 0.73.

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5594 - accuracy: 0.7258
Loss: 0.5594281554222107, Accuracy: 0.7258309125900269
```

*Optimisation 3.* Changed the activation function of the hidden layers from Relu to Tanh and used four hidden layers as below:

```
# Define the model - deep neural net
number_input_features = len(X_train[0])
hidden_nodes_layer1 =  100
hidden_nodes_layer2 = 50
hidden_nodes_layer3 = 50

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="tanh")
)

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="tanh"))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation="tanh"))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 100)               4400
_________________________________________________________________
dense_15 (Dense)             (None, 50)                5050
_________________________________________________________________
dense_16 (Dense)             (None, 50)                2550
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 51
=================================================================
Total params: 12,051
Trainable params: 12,051
Non-trainable params: 0
_________________________________________________________________
```

The accuracy is still 0.73 with a loss of 0.56.

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5582 - accuracy: 0.7252
Loss: 0.5582464337348938, Accuracy: 0.7252478003501892
```

## Summary

Attempts were made to increase the accuracy of the model by changing the number of hidden layers and the nodes in each layer, as well as changing the input activation function from Relu to Tanh. However, all the optimisation steps still resulted to about 73% accuracy.

It is possible that more data are needed to be able to determine what contributes to the success of a venture that can be used to create a predictive model.