Neural Network Model Analysis Report

Overview

- Client

The client for this project is the nonprofit foundation, Alphabet Soup.

- Purpose

The purpose of this project is to create a binary classifier that can predict whether applicants will be successful if backed by Alphabet Soup's funding. This will allow Alphabet Soup to select applicants for funding with the best chance of success in their ventures.

Results

- Data Preprocessing

What variable(s) are the target(s) for your model?

The target variable for the model is the "IS_SUCCESSFUL" column.

What variable(s) are the features for your model?

The features for the model include all the variables included in the data other than "IS_SUCCESSFUL", and the two columns that were initially dropped in preprocessing, "EIN" and "NAME". These features include "APPLICATION_TYPE", "AFFILIATION", "CLASSIFICATION", "USE_CASE, ORGANIZATION", "STATUS", "INCOME_AMT", "SPECIAL_CONSIDERATIONS", "ASK_AMT".

What variable(s) should be removed from the input data because they are neither targets nor features?

The variables that should be removed from the input data are "EIN" and "NAME" because they are indicators of each entry rather than features.

- Compiling, Training, and Evaluating the Model

How many neurons, layers, and activation functions did you select for your neural network model, and why?

The neural network model utilizes three layers because an ideal starting point for NNM's is 2-4 layers. There are 43 features so 100 neurons, or 2-3 times the amount of input features were used for the first hidden node. The second and third hidden nodes used 60 and 40 neurons respectively. The reLU activation function was utilized because it is ideal for modeling positive, nonlinear input data for classification or regression. The sigmoid function was utilized because its values are normalized to a probability between 0 and 1, which is ideal for a binary classification dataset.

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_23 (Dense)            (None, 80)                3520

 dense_24 (Dense)            (None, 50)                4050

 dense_25 (Dense)            (None, 40)                2040

 dense_26 (Dense)            (None, 1)                 41

=================================================================
Total params: 9,651
Trainable params: 9,651
Non-trainable params: 0
_________________________________________________________________
```

Were you able to achieve the target model performance?

The first model had an accuracy score of 72.7%.

```
268/268 - 1s - loss: 0.5607 - accuracy: 0.7261 - 833ms/epoch - 3ms/step
Loss: 0.5607346296310425, Accuracy: 0.726064145565033
```

After increasing the number of neurons and layers, the second model had an accuracy score of 72.5%.

```
268/268 - 0s - loss: 0.5786 - accuracy: 0.7249 - 460ms/epoch - 2ms/step
Loss: 0.5785872340202332, Accuracy: 0.7248979806900024
```

After increasing the epochs, the third model also had an accuracy score of 72.5%.

```
268/268 - 0s - loss: 0.5782 - accuracy: 0.7245 - 451ms/epoch - 2ms/step
Loss: 0.5781909227371216, Accuracy: 0.7245481014251709
```

After increasing the number of neurons, layers and epochs again, the fourth model had an accuracy score of 72.6%.

```
268/268 - 0s - loss: 0.5852 - accuracy: 0.7259 - 443ms/epoch - 2ms/step
Loss: 0.5851919054985046, Accuracy: 0.7259474992752075
```

These models did not reach the target model performance of 75%.

What steps did you take in your attempts to increase model performance?

To increase model performance, the model's neurons, layers and epochs were changed. In the second attempt at modeling, the neurons in each hidden layer were increased and a fourth hidden reLU layer was added. Adding more neurons speeds up the model and may reduce loss. Adding more layers considers more interactions between variables.

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 100)               4400

 dense_28 (Dense)            (None, 80)                8080

 dense_29 (Dense)            (None, 60)                4860

 dense_30 (Dense)            (None, 40)                2440

 dense_31 (Dense)            (None, 1)                 41

=================================================================
Total params: 19,821
Trainable params: 19,821
Non-trainable params: 0
_________________________________________________________________
```

In the third attempt at modeling, the epochs were increased from 100 to 200. As the number of epochs increases, so does the amount of information provided to each neuron. Adding more epochs also increases likelihood that model will achieve optimal weight coefficient.

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_42 (Dense)            (None, 100)               4400

 dense_43 (Dense)            (None, 80)                8080

 dense_44 (Dense)            (None, 60)                4860

 dense_45 (Dense)            (None, 40)                2440

 dense_46 (Dense)            (None, 1)                 41

=================================================================
Total params: 19,821
Trainable params: 19,821
Non-trainable params: 0
_________________________________________________________________
```

In the fourth attempt, the model's neurons, layers and epochs were increased. A fifth reLU layher was added and the epochs were increased from 200 to 250.

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_47 (Dense)            (None, 120)               5280

 dense_48 (Dense)            (None, 90)                10890

 dense_49 (Dense)            (None, 80)                7280

 dense_50 (Dense)            (None, 60)                4860

 dense_51 (Dense)            (None, 50)                3050

 dense_52 (Dense)            (None, 1)                 51

=================================================================
Total params: 31,411
Trainable params: 31,411
Non-trainable params: 0
_________________________________________________________________
```

Summary

- Model Overview

This deep learning model aimed to predict if a company would be classified as successful or no successful based on features of their application. Out of the four models, the highest accuracy score was 72.7% with the lowest loss of 58.5%. These results are not accurate enough for the clients threshold of 75% so more modeling attempts with different hyperparameters would need to be built to create a more reliable binary classifier.

- Additional Model Recommendation

Another model that could solve this classification problem is the Perceptron or linear binary classifier. The perceptron model mimics a biological neuron by receiving input data, weighting the information, and producing a clear output. It would be a good alternative method for classification because the model separates and classifies the data into two groups using linear equation. For the purpose of this project, those two groups would be successful and not successful.