Neural Network Model Report

Overview

Alphabet Soup, a nonprofit foundation, aims to optimize its funding allocation by leveraging machine learning and neural networks. Armed with a dataset encompassing 34,000+ organizations that have previously received funding, the task is to craft a binary classifier. This tool will analyze various features within the dataset, including identification markers like EIN and NAME, alongside critical organizational aspects such as APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION type, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, ASK_AMT, and IS_SUCCESSFUL. The objective? To predict the likelihood of success for potential ventures if funded by Alphabet Soup, thereby enhancing the foundation's decision-making process for future funding initiatives.

Results

Data Preprocessing

- What variable(s) are the target(s) for your model?
    - The IS_SUCCESSFUL variable is the target for my model which tells us if the company's past funding was successful.
- What variable(s) are the features for your model?
    - IS_SUCCESSFUL
- 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 neither targets nor features.

Compiling, Training, and Evaluating the Model

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

```
Model: "sequential_2"

 Layer (type)                    Output Shape                  Param #
 =================================================================
 dense (Dense)                   (None, 80)                    3760

 dense_1 (Dense)                 (None, 30)                    2430

 dense_2 (Dense)                 (None, 1)                     31


 =================================================================
Total params: 6221 (24.30 KB)
Trainable params: 6221 (24.30 KB)
Non-trainable params: 0 (0.00 Byte)
```

- Were you able to achieve the target model performance?

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

          268/268 - 1s - loss: 0.5558 - accuracy: 0.7240 - 536ms/epoch - 2ms/step
          Loss: 0.5557742714881897, Accuracy: 0.7239649891853333
```

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

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
n_input_features = len(X_train_scaled[0])
nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=80, activation="relu", input_dim = n_input_features))
# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=30, activation="relu"))
# Output layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))
# Check the structure of the model
nn.summary()
```

```
Model: "sequential_2"

_____
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                3760

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 6221 (24.30 KB)
Trainable params: 6221 (24.30 KB)
Non-trainable params: 0 (0.00 Byte)
_____
```

Summary

Alphabet Soup utilized machine learning and neural networks on a dataset of 34,000+ funded organizations to refine funding allocation. Their aim was to create a model predicting venture success if funded by Alphabet Soup, using the IS_SUCCESSFUL variable as the target, signifying past funding success. The model consisted of 80 neurons across 3 layers with ReLU activation but fell short of the 75% accuracy target, reaching 72.4%. There's a need to fine-tune without overfitting, highlighting the importance of balance in adjustments. The best model variation achieved a 72.7% accuracy using different neuron counts and layer configurations. Considering the random forest classifier's resilience to outliers, it's suggested for the next steps. To improve, reducing epochs between 20-50 is recommended.