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?

    - 80 neurons, 3 layers, and a ReLU activation function were selected for the neural network model because the number of features were dictated the number of hidden nodes as shown below:

```
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?
    - No, I was not able to achieve the target model performance of 75%. My attempt only garnered a 72.4% accuracy, which is still far from the desired goal.

```
0s [31] # 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 - 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?
    - To make the model better, I increased the neurons in a part of the network. This helps it understand more complex patterns in the data, like how things are connected. Also, I made it learn more by giving it more chances to adjust its settings through more "epochs," or learning cycles. This helps it get better at predicting things accurately. But I had to be careful not to do this too much, because too many cycles could make the model too focused on the data it already knows and not good at predicting new stuff. So, I balanced it to keep things working well.

```python
# 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

The Alphabet Soup foundation employed machine learning and neural networks to optimize funding allocation based on a dataset comprising 34,000+ previously funded organizations. The objective was to develop a binary classifier to predict the success likelihood of potential ventures if funded by Alphabet Soup, aiming to refine future funding decisions. The IS_SUCCESSFUL variable served as the model's target, signifying the success of past funding. Features included various organizational aspects, and the model incorporated 80 neurons across 3 layers with ReLU activation. Despite efforts to boost accuracy by increasing neuron count and learning cycles, achieving the 75% target remained elusive. The model yielded a 72.4% accuracy, indicating room for improvement, as excessive training cycles risked overfitting. Balancing these adjustments was crucial to maintain a functional predictive capacity while avoiding overreliance on existing data.