Neural Network Model Report

Overview - Purpose:

The primary goal of this analysis is to develop a binary classification tool for the nonprofit foundation Alphabet Soup, utilizing machine learning and neural network techniques. This tool will analyze historical data from over 34,000 organizations that have previously received funding, to predict the likelihood of an applicant's success in utilizing the funds effectively. The dataset includes various metadata about each organization, such as application type, industry affiliation, organizational classification, funding use case, organization type, active status, income classification, special considerations, and requested funding amount. By accurately predicting the potential success of funding applicants, Alphabet Soup can make more informed decisions, ensuring that their resources are allocated to ventures with the highest chance of making a positive impact.

Results:

Data Preprocessing:

a. **Target Variable**: The target variable for your model is "IS_SUCCESSFUL". This is what the model aims to predict: whether an applicant will be successful if funded by Alphabet Soup.
b. **Feature Variables**: The features for the model include all the other variables in the dataset except for "IS_SUCCESSFUL", "EIN", and "NAME". The features are "APPLICATION_TYPE", "AFFILIATION", "CLASSIFICATION", "USE_CASE", "ORGANIZATION", "STATUS", "INCOME_AMT", "SPECIAL_CONSIDERATIONS", and "ASK_AMT".
c. **Variables to be Removed**: The variables "EIN" and "NAME" should be removed from the input data as they are identification columns and do not contribute to the predictive power of the model.

Compiling, Training, and Evaluating the Model:

d. **Neurons, Layers, and Activation Functions:** For the neural network model, I selected 100 neurons for the first hidden layer and 250 neurons for the second hidden layer. The input layer was determined by the number of features in the dataset. For activation functions, I used "relu" for the first hidden layer and "sigmoid" for both the second hidden layer and the output layer.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.

number_input_features = X_train.shape[1]
hidden_nodes_layer1 =  100
hidden_nodes_layer2 = 250

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="sigmoid"))


# Output layer

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


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

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_30 (Dense)            (None, 100)               4500

 dense_31 (Dense)            (None, 250)               25250

 dense_32 (Dense)            (None, 1)                 251
```

e. **Target Model Performance:** The model achieved an accuracy of approximately 74.20% on the training data over 100 epochs. However, when evaluated on the test data, the accuracy dropped to 72.51%. This indicates that while the model learned the training data well, there was a slight decrease in performance which could be related to overfitting.

```
Epoch 93/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5311 - accuracy: 0.7414
Epoch 94/100
804/804 [==============================] - 2s 2ms/step - loss: 0.5312 - accuracy: 0.7417
Epoch 95/100
804/804 [==============================] - 2s 2ms/step - loss: 0.5308 - accuracy: 0.7416
Epoch 96/100
804/804 [==============================] - 2s 3ms/step - loss: 0.5310 - accuracy: 0.7423
Epoch 97/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5306 - accuracy: 0.7427
Epoch 98/100
```

```
l02] # 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: 0.5725 - accuracy: 0.7251 - 430ms/epoch - 2ms/step
Loss: 0.5724639296531677, Accuracy: 0.7251312136650085
```

f. **Steps to Increase Model Performance:** To improve the model's performance, I adjusted the number of neurons in each layer. After experimenting with various configurations, I found that optimizing the neuron count in each layer minimally impacted the model's accuracy. Additionally, I selected the sigmoid activation function for both the second and output layers. This decision was based on the sigmoid function's suitability for binary classification tasks (1 or 0 in our case).

Summary:

In conclusion, the model demonstrates a satisfactory level of performance in forecasting the successful utilization of funds by applicants, achieving an accuracy rate of approximately 72%. However, the loss percentage reaches the 57%, indicating room for improvement in model efficiency. To further refine and optimize the model, a systematic approach involving the exploration of various network architectures could be beneficial. This includes experimenting with diverse configurations of neurons, layers, and activation functions to identify the most effective combination for this specific task. If needed, using less iterations to terminate training when the validation loss stops decreasing can prevent overfitting.