Deep Learning Challenge – Predict Applicant's Performance

Overview:

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With your knowledge of machine learning and neural networks, you'll use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

Results:

- Data Preprocessing:
  The target for my model: IS_SUCCESSFUL
  The features for my model:
  NAME, APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, ASK_AMT
  The variable which had been removed: EIN
  (NAME was removed at first but then added back to the model, since it is relevant information, accuracy was increased after adding it back)
- Compiling, Training, and Evaluating the Model:

```
[ ] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
    number_input_features = len(X_train_scaled[0])
    hidden_nodes_layer1 = 8
    hidden_nodes_layer2 = 16

    nn = tf.keras.models.Sequential()

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

    # 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()
```

This is my first attempt with the 2 layers and activation function selected for my neural network model. The accuracy of my first attempt was a little more than 73% but under 74% with just 2 layers.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 7
hidden_nodes_layer2 = 14
hidden_nodes_layer3 = 21

nn = tf.keras.models.Sequential()

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

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

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

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

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

Here is the layers, activation function I finally selected for my neural network model, since these selections had the best outcome. I was able to achieve the target model performance. I got 78.94%.

```
# 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: 0.4680 - accuracy: 0.7894 - 341ms/epoch - 1ms/step
Loss: 0.4679853320121765, Accuracy: 0.7893877625465393
```

In my attempts to increase model performance: I tried to change the layers, add more layers, change the cutoff value for the variable, add and reduce the number of epochs to the training regimen and drop/add a column.

Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 3675

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 21)                315

 dense_3 (Dense)             (None, 1)                 22

=================================================================
Total params: 4,124
Trainable params: 4,124
Non-trainable params: 0
_________________________________________________________________
```

Total params is 4124 with a 3-layers model is the best model. My accuracy is 78.94%. As a result, choosing multiple layers and choosing the right features could help to build a better model and solve this classification problem.