Neural Network Model Report

Overview: The purpose of the analysis was to evaluate and include data useful in predicting if applicants will be successful if funded by Alphabet Soup.

Results: As part of the Data Preprocessing, I evaluated the variables provided in the dataset.

Target variable: "IS_SUCCESSFUL" was my target variable as the field was determined if the money funded was used effectively by the organizations.

Feature variables: The following features were used in my model: APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, INCOME_AMT, and ASK_AMT

For one of the models, I removed the "ASK_AMT" variable, which was the Funding amount requested, to see if that variable with the wide range of responses was causing issues. After I ran it (Optimization #2) found that it was not helpful for that process.

Variables removed: Initially we removed two identification variables, both EIN and NAME, as they were both for identification.

For a couple models I also removed two variables; STATUS and SPECIAL CONSIDERATIONS, as the large majority of respondents had the same, with limited variation.

Compiling, Training, and Evaluating the Model:

I initially selected 80 neurons for 1st hidden layer and 30 for the 2nd hidden layer. I did the same for the first two optimization models while make other adjustments. In the third optimization I changed it to 40 neurons for in layer 1 and 15 in layer 2, but that made accuracy work.

In the initial and all three optimization models, I used "relu" for the first and second activation functions, and "sigmoid" for the Output layer.

I was not able to achieve the target model performance. The first model had accuracy of 72% with 56% loss.

First optimization I dropped both the STATUS and SPECIAL CONSIDERATIONS variables as they did not seem necessary due to the percentage of each. I also increased to 150 epochs for testing. Accuracy went up slightly, closer to 73% but otherwise the same.

Second optimization I reduced back to 100 epochs and removed the ASK_AMT variable due to high variation of number, but results were similar.

For the third optimization I put the ASK_AMT back in changed neurons in layer1 to 40 and layer2 down to 15. Results still didn't vary much but accuracy dropped a little.

Summary: The goal of the model and optimization were to hit an accuracy of 75% in predicting if applicants would be successful in determining the affect of funding by Alphabet Soup. A couple of my models looked close when they were running but never raised up to that 75% goal.

Closer evaluation of the columns in the dataset may have helped in eliminating more or possibly better defining the primary determinants. I would also try a different activation function to solve.

Initial model:

```
# Define the model - deep neural net, i.e.,
number_input_features = len(X_train[0])
hidden_nodes_layer1 =  80
hidden_nodes_layer2 = 30
```

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

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 5981 (23.36 KB)
Trainable params: 5981 (23.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Train the model
fit_model = nn.fit(X_train_scaled,y_train,epochs=100)
```

```
Epoch 1/100
804/804 [==============================] - 3s 2ms/step - loss: 0.5700 - accuracy: 0.7242
Epoch 2/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5549 - accuracy: 0.7321
Epoch 3/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5516 - accuracy: 0.7330
```

```
Epoch 99/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5342 - accuracy: 0.7411
Epoch 100/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5340 - accuracy: 0.7421
```

```
[17] # 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: 0.5584 - accuracy: 0.7243 - 515ms/epoch - 2ms/step
Loss: 0.5584347248077393, Accuracy: 0.7243148684501648
```

1st optimization:

```
46] # 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: 0.5679 - accuracy: 0.7259 - 467ms/epoch - 2ms/step
Loss: 0.5678615570068359, Accuracy: 0.7259474992752075
```

2nd optimization:

```
# 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: 0.5607 - accuracy: 0.7234 - 345ms/epoch - 1ms/step
Loss: 0.5606633424758911, Accuracy: 0.7233819365501404
```

```
# Export our model to HDF5 file
from tensorflow.keras.models import save_model
save_model(nn, "AlphabetSoupCharity_Optimization2.h5")
```

3rd optimization:

```
# Define the model - deep neural net, i.e., the number of input features
number_input_features = len(X_train[0])
hidden_nodes_layer1 =  40
hidden_nodes_layer2 = 15

nn = tf.keras.models.Sequential()

# First hidden layer
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 40)                1640

 dense_4 (Dense)             (None, 15)                615

 dense_5 (Dense)             (None, 1)                 16

=================================================================
Total params: 2271 (8.87 KB)
Trainable params: 2271 (8.87 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Epoch 28/100
804/804 [==============================] - 2s 2ms/step - loss: 0.5414 - accuracy: 0.7379
Epoch 29/100
804/804 [==============================] - 1s 2ms/step - loss: 0.5409 - accuracy: 0.7380
```

```
[22] # 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: 0.5589 - accuracy: 0.7243 - 359ms/epoch - 1ms/step
Loss: 0.5588815212249756, Accuracy: 0.7243148684501648
```

```
# Export our model to HDF5 file
```