Optimizing Alphabet Soup

Overview: The purpose of this analysis is to create a binary classifier that can predict whether an applicant will be successful if funded by Alphabet Soup. This analysis will help Alphabet Soup make better, more informed decisions when deciding which applicants to approve.

Results:

Data Preprocessing:

The target variable in the model is whether the application “IS_SUCCESSFUL” shown by a binary score in the “IS_SUCCESSFUL” column of the dataset.

The feature variables are the application type, the affiliated sector, the government organization classification, the use case for funding, the organization type, the status, the income amount, the ask amount, and special considerations. These are identified by the following columns. ['APPLICATION_TYPE', 'AFFILIATION', 'CLASSIFICATION', 'USE_CASE', 'ORGANIZATION', 'INCOME_AMT', 'SPECIAL_CONSIDERATIONS']

There were two identification columns “EIN and NAME” that were removed from the dataset as they were neither targets nor features.

Compiling, Training, and Evaluating the Mode:

I had 3 layers (2 hidden, and output layer). The activation function are RELU on the first two and sigmoid on the output layer. The first layer has 8 neurons and the second layer has 5 neurons. These choices were made due to experimentation with different numbers and layers and this was the highest accuracy combination.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_9 (Dense)              (None, 8)                 520

 dense_10 (Dense)             (None, 5)                 45

 dense_11 (Dense)             (None, 1)                 6

=================================================================
Total params: 571
Trainable params: 571
Non-trainable params: 0
_________________________________________________________________
```

While close, I was not able to reach target model performance as the accuracy score was 72.46%. I created 3 different attempts to optimize my model. In my first attempt, I did not filter out values out of the application and classification types. The resulting accuracy of this model was slightly lower at 72.33%.In the second attempt, I increased the neurons in my first and second hidden layer from 8 and 5 to 10 and 12. The resulting accuracy of this model was slightly higher at 72.51%.In my third attempt, I added an additional hidden layer with 14 neurons and sigmoid activation layer.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                1170

 dense_1 (Dense)             (None, 12)                132

 dense_2 (Dense)             (None, 14)                182

 dense_3 (Dense)             (None, 1)                 15

=================================================================
Total params: 1,499
Trainable params: 1,499
Non-trainable params: 0
_________________________________________________________________
```

The resulting accuracy of this model was slightly higher at 72.72%.

```
804/804 [==============================] - 2s 4ms/step - loss: 0.5440 - accuracy: 0.7
```

In [10]:
```
# 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: 0.5554 - accuracy: 0.7272 - 1s/epoch - 5ms/step
Loss: 0.5553853511810303, Accuracy: 0.7272303104400635
```

Summary: Overall, I got the highest accuracy level during my third attempt. A model with more hidden layers could possibly provide a higher accuracy score. This is recommended based on the fact that this dataset has a high number of features.