Overview:

The purpose of this analysis is to evaluate a binary classifier created by us to predict whether applicants will be successful if funded by Alphabet Soup. The binary classifier was created through training from a given csv with over 340,00 organizations that have received funding from Alphabet soup.

Results:

Data Preprocessing:

- Q. What variable(s) are the target for your model?

  A. The target variable was the "IS_SUCCESFUL" column from the given CSV.

- Q. What variable(s) are the features for your model?

  A. The feature variables for our model were the 43 columns in our preprocessed data frame, not including the "IS_SUCCESFUL" column.

- Q. What variable(s) should be removed from the input data because they are neither targets nor features?

  A. The two variables that were removed from due to not being a target nor feature were the "EIN" and "NAME" columns.

Compiling, Training, and Evaluating the Model:

- Q. How many neurons, layers, and activation functions did you select for your neural network model, and why?

  A. For the original neural network model, there were 2 hidden layers and 1 output layer, the first and second hidden layer both used the relu activation function and had 8 and 6 nodes respectively. The output layer used the sigmoid function and had 1 node. They way the model was setup allowed for variability for number of layers and nodes and function is needed.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_feat = len(X_train_scaled[0])
hidden_nodes_1 = 8
hidden_nodes_2 = 6

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_1, activation="relu", input_dim=input_feat))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_2, activation="relu"))

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

- Q. Were you able to achieve the target model performance?

  A. Sadly no, even after optimization the highest accuracy score achieved was 72.94%.

- Q. What steps did you take in your attempts to increase model performance?

  A. Three optimizations were made to increase model performance. The first optimization consisted of increasing the number of nodes for the first and second layer to 22 and 10 respectively.

```
hidden_nodes_1: 8 --> 22

hidden_nodes_2: 6 --> 10

Accuracy: 0.7282 --> 0.7266
```

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_feat = len(X_train_scaled[0])
hidden_nodes_1 = 22
hidden_nodes_2 = 10


nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_1, activation="relu", input_dim=input_feat))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_2, activation="relu"))

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

The second optimization consisted of adding a new hidden layer with 6 nodes and the tahn function.

hidden_nodes_1: 8 --> 22

hidden_nodes_2: 6 --> 10

hidden_nodes_3: 6

Accuracy: 0.7266 --> 0.7272

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_feat = len(X_train_scaled[0])
hidden_nodes_1 = 22
hidden_nodes_2 = 10
hidden_nodes_3 = 6

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_1, activation="relu", input_dim=input_feat))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_2, activation="relu"))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_3, activation="tanh"))

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

The final optimization consisted of adding another hidden layer with the tahn function and 20 nodes. I also adjusted the number of nodes for the first three layers to 8,12,16 respectively. It should be noted that unlike the other models, the number nodes increase with every new hidden layer.

hidden_nodes_1: 22 --> 8

hidden_nodes_2: 10 --> 12

hidden_nodes_3: 6 --> 16

hidden_nodes_4: 20

Accuracy: 0.7272 --> 0.7294

Sadly could not break the 73% mark for accuracy.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_feat = len(X_train_scaled[0])
hidden_nodes_1 = 8
hidden_nodes_2 = 12
hidden_nodes_3 = 16
hidden_nodes_4 = 20

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_1, activation="relu", input_dim=input_feat))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_2, activation="relu"))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_3, activation="tanh"))

# Fourth hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_4, activation="tanh"))

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

Summary:

The original model had an accuracy of 72.82% and after three optimizations actually decreased to 72.72%. Thus, the target model performance of 75% was not achieved. I would recommend using different a classification models instead of Sequential to potentially boost the accuracy score. Potentially one that took less optimization to relate the input data to the output results.