Analysis

- **Overview:**
  - This project tasked us with reviewing data for a fictional nonprofit foundation Alphabet Soup. This company wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With our knowledge of machine learning and neural networks, we used 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:**
    - An "X" and "y" variable were created to identify our features and target arrays. In addition, EIN and Name columns were correctly removed as instructed for this dataset.
  - **Compiling, Training, and Evaluating the Model**
    - The initial attempt for the Neural Network was set to two initial layers. Our optimization attempt included two additional layers; however, the desired accuracy of 75% could not be obtained. The final accuracy obtained was 73.43%.

```
# 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 = 9
hidden_nodes_layer2 = 18

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="relu"))

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

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 9)                 396

 dense_1 (Dense)             (None, 18)                180

 dense_2 (Dense)             (None, 1)                 19

=================================================================
Total params: 595
Trainable params: 595
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] # 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: 0.5456 - accuracy: 0.7336 - 952ms/epoch - 4ms/step
    Loss: 0.5456297397613525, Accuracy: 0.7336443066596985
```

```
[20] # 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
     hidden_nodes_layer3 = 24
     hidden_nodes_layer4 = 32

     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="relu"))

     nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation="relu"))

     nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer4, activation="relu"))

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

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

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 8)                 352

 dense_12 (Dense)            (None, 16)                144

 dense_13 (Dense)            (None, 24)                408

 dense_14 (Dense)            (None, 32)                800

 dense_15 (Dense)            (None, 1)                 33
```

```
[23] # 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: 0.5525 - accuracy: 0.7343 - 536ms/epoch - 2ms/step
     Loss: 0.5524836778640747, Accuracy: 0.7343440055847168
```

- **Summary:**
  - The existing layers within the optimization attempt would need additional fine tuning to reach the desired accuracy output of 75%.