Deep Learning Challenge

In this analysis we utilized deep learning and neural networks to assess the likelihood of successful funding for applicants by Alphabet Soup. From Alphabet Soup's business team, you have received a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years.

Data Processing:

I determined the target variable(s) and feature variable(s) for the model, dropping the irrelevant "EIN" and "NAME" columns. I calculated the number of unique values in each column and used this information to bin infrequent categorical variables into an "Other" category. I encoded categorical variables using pd.get_dummies() and split the preprocessed data into training and testing datasets. Finally, I scaled the features using StandardScaler to ensure consistent data representation.

Compiling, Training, and Evaluating the Model:

A neural network was employed, comprising two layers as seen below.

```
# 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 = 20

nn = tf.keras.models.Sequential()

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

# 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 training model produced 553 parameters. The initial attempt achieved a close accuracy of 72%, which fell slightly below the desired 75%.

```
268/268 - 0s - loss: 0.5514 - accuracy: 0.7273 - 445ms/epoch - 2ms/step
Loss: 0.5514331459999084, Accuracy: 0.7273469567298889
```

Optimization:

In the second attempt, I adjusted the node layers and decreased the epochs to 15. This model produced 613 parameters with accuracy of 72% again.

```
268/268 - 0s - loss: 0.5518 - accuracy: 0.7262 - 450ms/epoch - 2ms/step
Loss: 0.5517914891242981, Accuracy: 0.7261807322502136
```

In my final attempt, I only used relu activation layers and increased the epochs to 40.

```
# 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 = 15
hidden_nodes_layer2 = 8

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='relu'))

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

```
268/268 - 0s - loss: 0.5909 - accuracy: 0.7283 - 446ms/epoch - 2ms/step
Loss: 0.5909460783004761, Accuracy: 0.7282798886299133
```

The accuracy did not reach the 75% target, but improved slightly.

Summary:

In summary, using relu and sigmoid activation layers did not yield the desired results. To improve upon the model, additional layers could be added with different activation layers. Additional layers could more accurately predict results due to the non-linearity of the data set. Alternatively, dropout layers could be added to reduce any over-fitting.