# Deep Learning Project

**Overview Of the Analysis:**
The purpose of this analysis is to create a deep learning model using a neural network to predict the success of funding applications for Alphabet Soup, a charitable organization. The model aims to classify whether an organization will be successful or not in receiving funding based on various features.

**Data Preprocessing:**
The features for the model include the following columns from the dataset: "APPLICATION_TYPE","AFFILIATION", "CLASSIFICATION", "USE_CASE", "ORGANIZATION", "STATUS", "INCOME_AMT", "SPECIAL_CONSIDERATIONS" and "ASK_AMT".
The "EIN" and "NAME" column is removed from the input data as it is neither a target nor a feature.

**Compiling, Training, and Evaluating the Model:**

- The neural network model consists of three layers: two hidden layers and an output layer.
- The first hidden layer has 9 neurons, the second hidden layer has 18 neurons, and the output layer has 1 neuron.
- The activation function used in the hidden layers is the ReLU (Rectified Linear Unit) activation function, which helps introduce non-linearity to the model.
- The output layer uses the sigmoid activation function to produce a binary classification output.
- The model was trained for 80 epochs with a validation split of 15%.
- The training model achieved an accuracy of 72% which was under the desired 75%.

```
#  YOUR CODE GOES HERE
input_features = len( X_train_scaled[0])
hidden_layer1=9
hidden_layer2=18
hidden_layer3=27
nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_layer1, input_dim=input_features, activation='relu'))

# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=hidden_layer2, activation='relu'))

# Output layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

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

```
1] # 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: 0.5524 - accuracy: 0.7262 - 367ms/epoch - 1ms/step
   Loss: 0.5523694157600403, Accuracy: 0.7261807322502136
```

Add code cell
⌘/Ctrl+M B

```
nn.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 9)                 450

 dense_1 (Dense)             (None, 18)                180

 dense_2 (Dense)             (None, 1)                 19

=================================================================
Total params: 649
Trainable params: 649
Non-trainable params: 0
_________________________________________________________________
```

This model created 649 parameters.

**Optimization:**

In the second model, “NAME” is added back to the data set. This time I achieved 78.7% which is approximately 4% over the target with 4276 parameters.

```
# 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_3 (Dense)             (None, 9)                 4077

 dense_4 (Dense)             (None, 18)                180

 dense_5 (Dense)             (None, 1)                 19

=================================================================
Total params: 4,276
Trainable params: 4,276
Non-trainable params: 0
_________________________________________________________________
```

The deep learning model performed reasonably well with an accuracy of 78.7% on the test data. It shows promising results in predicting the success of funding applications based on the provided features.

```python
# 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: 0.4698 - accuracy: 0.7871 - 355ms/epoch - 1ms/step
Loss: 0.4698288142681122, Accuracy: 0.7870553731918335
```