Overview:

A non profit foundation Alphabet Soup wants to create an algorithm hat predicts whether or not the applicants in the dataset will be successful. With machine learning and neural networks, w must use the feature in the provided dataset to create a binary classifier that is capable of predicting whether applicants will be successful if funded by Alphabet Soup. This dataset had 34,000 organizations and 12 columns and it captured the metadata about each organzaization and their past funding outcomes.

Data Preprocessing:

- What variables are the targets for your model??
    - Is_Successful is the target that is marked 1 for successful and 0 for unsuccessful.

- What variables are the features for your model??
    - Is_Successful, is the feature column chosen data for the model

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

Compiling, Training, and Evaluating the Model

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

There were 3 layers total for each model after applying Neural Networks. The number of hidden nodes were dictated by the number of features

```
# 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=7

hidden_nodes_layer2=14

hidden_nodes_layer3=21
```

```
nn = tf.keras.models.Sequential()

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_1"
_________________________________________________________________

Layer (type) Output Shape Param #
=================================================================

dense (Dense) (None, 7) 350

dense_1 (Dense) (None, 14) 112

dense_2 (Dense) (None, 1) 15
================================================================= Total
params: 477 (1.86 KB) Trainable params: 477 (1.86 KB) Non-trainable
params: 0 (0.00 Byte)

268/268 - 1s - loss: 0.5531 - accuracy: 0.7289 - 501ms/epoch - 2ms/step
Loss: 0.5530825257301331, Accuracy: 0.728863000869751
```

477 parameters were created by a three-layer training model the first attempt was %72.88 which was just slightly under our target score of 75%.

```
# 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=7

hidden_nodes_layer2=14

hidden_nodes_layer3=21

nn = tf.keras.models.Sequential()

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()
```

Optimization

Model: "sequential_1"

_____

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	3171
dense_1 (Dense)	(None, 14)	112
dense_2 (Dense)	(None, 1)	15

==========

Total params: 3298 (12.88 KB)

Trainable params: 3298 (12.88 KB)

Non-trainable params: 0 (0.00 Byte)

_____

```
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: 0.4728 - accuracy: 0.7880 - 566ms/epoch - 2ms/step
Loss: 0.47283393144607544, Accuracy: 0.7879883646965027
```

To achieve the target model performance?

The target for the model achieved a 78% model performance when we chose to optimize our code.We added the “Name” column which increased our parameters into our model and achieved an accuracy of 78% and nearly 79%. We chose a different cutoff value and looked to create a list of classsifications that were to be replaced and then binned. We used multiple layers because we wanted to classify information based on filtering inputs through layers.

Summary:

The models best result employing the different number of neurons and layers was 78%. Another recommendation that I could suggest to help solve this classification problem could be to reduce our number of epochs to around the 30-60 range. This would reduce our outliers and could help us achieve a better classification.