Overview:
Neural networks and machine learning is used to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

Data Preprocessing:
The EIN and NAME column were dropped, since they contained irrelevant information. The "IS_SUCCESSFUL" column was used as the target variable and all other columns were used as features for the model.

Compiling, Training, and Evaluating the Model:
A three layer training model was used because deep learning models should have multiple layers. It teaches a computer to filter inputs through the layers to learn how to predict and classify information.The second attempt added the 'Name' column back into the dataset in order to increase the model's performance. This second attempt was success.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
# Define the model - deep neural net
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()

# 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_3 (Dense)              (None, 80)                3520
_________________________________________________________________
dense_4 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 31
=================================================================
Total params: 5,981
Trainable params: 5,981
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 - 0s - loss: 0.5548 - accuracy: 0.7250
Loss: 0.5547791123390198, Accuracy: 0.7250145673751831
```

Summary:
In my second attempt I was able to achieve 79% accuracy in comparison to 73% in my first attempt. A support vector machine model could have also been used, since it is also efficient with binary classification problems.

```
# 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: 0.4455 - accuracy: 0.7932 - 250ms/epoch - 932us/step
Loss: 0.4454589784145355, Accuracy: 0.7932361364364624
```