OVERVIEW

The purpose of this challenge was to use machine learning and neural networks to create a binary classifier using a given dataset that can predict whether applicants will be successful if they are funded by the nonprofit foundation Alphabet Soup.

RESULTS

Data Preprocessing:

- For this challenge, we imported and read a CSV containing more than 34,000 organization that have received funding from Alphabet Soup.
    - Basic cleaning of the CSV was done, such as dropping unnecessary columns ("EIN" & "NAME"), which were unnecessary.
- 'IS_SUCCESSFUL' was set as the target variable.
- 'CLASSIFICATION' was a value used for binning.
- train_test_split was used to split the preprocessed data into a training and testing dataset.
- There was two hidden layers that were created, with one outer layer. The nodes were 80, 30, and 1 respectively.

```
[ ] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
    #  YOUR CODE GOES HERE
    number_input_features = len(X_train_scaled[0])
    nn = tf.keras.models.Sequential()

    # First hidden layer
    #  YOUR CODE GOES HERE
    nn.add(tf.keras.layers.Dense(units=80, input_dim=number_input_features, activation='relu'))
    # Second hidden layer
    #  YOUR CODE GOES HERE
    nn.add(tf.keras.layers.Dense(units=30, 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()
```

    - Relu was used because of its reliability and simplicity.
    - Sigmoid was used for the outlayer due to it being suitable for binary classifications.
- After several attempts and tweaks, the accuracy rate gotten was 68%, which was the closest to the goal of 75%.

```
[ ] # 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}")

    215/215 - 0s - loss: 0.8439 - accuracy: 0.6863 - 419ms/epoch - 2ms/step
    Loss: 0.8438572883605957, Accuracy: 0.6862973570823669
```

SUMMARY

68% Accuracy was the highest score gotten. In an alternative ipynb file named AlphabetSoupCharity_Optimization was created. There was an additional hidden layer added, with the original nodes increased, except for the outer layer. The activation methods were also changed.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE
number_input_features = len(X_train_scaled[0])
nn = tf.keras.models.Sequential()

# First hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=128, input_dim=number_input_features, activation='relu'))
# Second hidden layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=64, activation='sigmoid'))
nn.add(tf.keras.layers.Dense(units=33, activation='relu'))
# Output layer
#  YOUR CODE GOES HERE
nn.add(tf.keras.layers.Dense(units=1, activation='softmax'))
# Check the structure of the model
nn.summary()
```

Epochs was first set as 100 with a result of 53%, and it then set to 200, with the result still being 53%.

```
[119] # 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}")

      215/215 - 0s - loss: 0.6928 - accuracy: 0.5341 - 411ms/epoch - 2ms/step
      Loss: 0.6928409337997437, Accuracy: 0.5341107845306396
```

The original ipynb was the one with the highest accuracy rate, even if it didn't achieve the desired 75% accuracy rate. Even after the completion of this challenge, I plan to continue on adjusting my code to be able to achieve the desired accuracy rate.