AlphabetSoup Analysis

Overview:
The purpose of this analysis was to use deep learning and neural networks tools to predict if applicants would be successfully funded by AlphabetSoup. The dataset included over 34,000 organizations.

Results:

- Data preprocessing

  The input dataset had 12 variables total, and the target model was “IS_ SUCCESSFUL” both ‘EIN’ and ‘NAME’ columns were dropped since it was considered irrelevant information, Leaving 9 feature variables. For binning both “APPLICATION_TYPE’ and “CLASSIFICATION’ were grouped together. For unique values “Rare” was put together in a new value “Other”. Then the categorical values were encoded using pd.get_dummies().

```
[18] # Choose a cutoff value and create a list of classifications to be replaced
     classifications_to_replace = list (valcountbinning[valcountbinning<100].index)

     # Replace in dataframe
     for cls in classifications_to_replace:
         application_df['CLASSIFICATION'] = application_df['CLASSIFICATION'].replace(cls,"Other")

     # Check to make sure binning was successful
     application_df['CLASSIFICATION'].value_counts()
```

```
C1000    17326
C2000     6074
C1200     4837
C3000     1918
C2100     1883
C7000      777
Other      669
C1700      287
C4000      194
C5000      116
C1270      114
C2700      104
Name: CLASSIFICATION, dtype: int64
```

```
[19] # Convert categorical data to numeric with `pd.get_dummies`
     application_df = pd.get_dummies(application_df,dtype=float)
     application_df.head()
```

- Compiling, Training and Evaluating the Model
  For the Neural Network there were 3 layers total and 2 hidden layers with an additional output layer. First hidden layer had 7 neurons and the second layer had 14 neurons. Each hidden layer used the ReLu activation function and the output layer used the Sigmoid activation function.

```
# 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

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_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 7)                 350

 dense_7 (Dense)             (None, 14)                112

 dense_8 (Dense)             (None, 1)                 15
```

After testing the model in the first attempt it generated 477 parameters with an accuracy of 72% which under performed from the target goal of 75%.

```
268/268 - 0s - loss: 0.5564 - accuracy: 0.7278 - 362ms/epoch - 1ms/step
Loss: 0.5564090013504028, Accuracy: 0.7278134226799011
```

To optimize the model to get better accuracy the ‘NAME’ variable was added back to the dataset. In result there were 701 parameters with an accuracy of 75% meeting the desired goal.

```
[27] Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 7)                 574

 dense_13 (Dense)            (None, 14)                112

 dense_14 (Dense)            (None, 1)                 15

=================================================================
Total params: 701
Trainable params: 701
Non-trainable params: 0
_________________________________________________________________
```

```
268/268 - 0s - loss: 0.5010 - accuracy: 0.7506 - 373ms/epoch - 1ms/step
Loss: 0.5010480284690857, Accuracy: 0.7505539655685425
```

- Summary
  Overall the deep learning model using multiple layers with additional variables gave the accuracy results of 75% allowing the nonprofit foundation to predict funding.