Overview

For this project, we created a binary classifier for the company Alphabet Soup to predict which of their applicants would be successful if funded. Data from over 34,000 unique applications was used to create these models.

Preprocessing

First, irrelevant data was dropped - this includes only the EIN column for the optimized model, as it is an ID value unique to each column and not important for any sort of chronological considerations. Initially, the NAME column was also dropped, but was added back in to improve model accuracy.

APPLICATION and CLASSIFICATION columns were binned by type, with the values with few results being grouped together in an “other” value. After these preprocessing steps, categorical data was converted into numeric data to be processed by the model.

The target for this model is the variable that denotes “good” applicants in the “IS SUCCESSFUL” column, which contains values of 1 for “no” and 0 for “yes”.

The features of this model are:
Application Type
Affiliation
Classification
Use Case
Type of Organization
Active Status
Income Amount
Special Considerations
Ask Amount

Compiling, Training, and Evaluating the Model

This model uses 3 total layers, 2,277 neurons, and the activation functions sigmoid and relu.

```
# 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 =  8
hidden_nodes_layer2 = 6

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()
```

[40]

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 8)                 2216
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 54
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 7
=================================================================
Total params: 2,277
Trainable params: 2,277
Non-trainable params: 0
_________________________________________________________________
```

The target model performance of 75%+ was achieved by adding the NAME column back into the model, which was originally removed. Other attempts included removing the STATUS and SPECIAL_CONSIDERATIONS columns, since there were only two possible values for each column and only a few rows had the less common values, however this decreased accuracy to 63%. I also tried adjusting the number of layers, and the changes to accuracy were negligible.

Summary

The optimized model including the NAME column was successful in reaching the target accuracy. To increase accuracy even further, the number of layers and nodes can be adjusted further, and different activation functions could be used.