Alphabet Soup Neural Network Report

Overview

The neural network constructed in the associated notebooks seeks to produce a reliable predictive model as it relates to the success of charity organizations.

The neural network aggregates several related features per organization to attempt to predict the organization's success as a binary decision (successful or not successful).

Results

Data Preprocessing

- The target of the model was the success of the organization as a binary outcome (successful or not successful).
- The features used by the model were the application type, sector of industry, government organization classification, case for funding, organization type, and funding amount requested.
- The identification and name of the organization were not considered, initially, as they should in theory hold no weight in determining success. However, as will be discussed later, the name was reintroduced as a categorical variable.

Compiling, Training, and Evaluation

- The current model contains three hidden layers. Layer one has a rectified linear unit (relu) activation with 15 neurons, layer two has an exponential linear unit (elu) activation with 20 neurons, and layer three has a relu activation with 20 neurons.

Layer (type)	Output Shape	Param #
dense_92 (Dense)	(None, 15)	765
dropout_41 (Dropout)	(None, 15)	0
dense_93 (Dense)	(None, 20)	320
dropout_42 (Dropout)	(None, 20)	0
dense_94 (Dense)	(None, 20)	420
dense_95 (Dense)	(None, 1)	21

- Target performance of 0.75 was able to be reached.

```
Epoch 7/100
515/515 [==============================] - 0s 910us/step - loss: 0.5241 - accuracy: 0.7418
Epoch 8/100
515/515 [==============================] - 0s 920us/step - loss: 0.5241 - accuracy: 0.7408
Epoch 9/100
515/515 [==============================] - 0s 905us/step - loss: 0.5229 - accuracy: 0.7425
Epoch 10/100
515/515 [==============================] - 0s 951us/step - loss: 0.5226 - accuracy: 0.7425
Epoch 11/100
515/515 [==============================] - 0s 907us/step - loss: 0.5206 - accuracy: 0.7435
Epoch 12/100
515/515 [==============================] - 0s 910us/step - loss: 0.5215 - accuracy: 0.7431
Epoch 13/100
...
Epoch 100/100
515/515 [==============================] - 0s 914us/step - loss: 0.5112 - accuracy: 0.7467
268/268 - 0s - loss: 0.5148 - accuracy: 0.7480 - 267ms/epoch - 996us/step
Loss: 0.5147939920425415, Accuracy: 0.7479883432388306
```

- Attempts were made to optimize performance through several means. First, removing special consideration, while also adding and rebinning organization names. This was done as it was determined through repeated testing that special consideration did not benefit the model accuracy, whereas the organization name was a value which was not unique unlike the identification number, this meant it was possible to rebin the names to add more depth to the data. Secondly, the number of neurons was reduced across all layers. This was done to balance efficiency as well as results. Finally, the number of hidden layers as well as activation functions were adjusted to account for the complexity of the data.
- Optimization 1:

```
## Optimization 1: Drop several columns with very unbalanced distributions
dropCols = ['EIN','SPECIAL_CONSIDERATIONS','INCOME_AMT']
application_df.drop(columns=dropCols,inplace = True,errors='ignore')

# Determine the number of unique values in each column.
application_df.nunique()
for i in application_df.columns:
    print(application_df[i].value_counts())
```

- Optimization 2:

```
# Compile, Train and Evaluate the Model
## Optimization 2: Change amount of neurons in each layer
number_input_features = X_train.shape[1]
hidden_nodes_layer1 = 15
hidden_nodes_layer2 = 20
hidden_nodes_layer3 = 20
```

- Optimization 3:

```
# First hidden layer
## Optimization 3: Add more layers and dropout layers to account for overfitting
nn.add(
    tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="relu")
)
nn.add(tf.keras.layers.Dropout(0.15))
# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="elu"))
nn.add(tf.keras.layers.Dropout(0.15))
# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation="relu"))
```

Summary

Overall, the neural networks performance is only good. It is not an exceptional predictor in its current form, but further tuning may reveal a more optimal set up. However, it may also be more beneficial to use a random forest classifier. The current dataset is quite small and therefore a simple machine learning model may prove optimal. Due to the binary nature of the problem, as well as several features which may have direct relationships to the target, a decision tree may be a more direct and efficient route.