Alphabet Soup for Charity!

Overview

The foundation Alphabet Soup wants us to create a model for predicting whether or not those asking for funding for charity will succeed, or reach the amount they ask for. Using neural networking and machine learning, we will create such a model for them!

Results

Data Preprocessing

To start, we need to understand what our target was in the dataset.”IS_SUCCESSFUL” was a binary column of data that said whether or not a charity succeeded in funding. The rest of the data was considered our features, though some useless data needed to be scrubbed. Namely the “EIN” column which was just numbers assigned to each fundraiser. “NAME” was also considered junk data originally, but later on it was learned there was value in it.”Application” and “CLASSIFICATION” columns were used for binning as not all of the data was as common as the rest, so the ‘rare’ data was put together in Other. Once the binning was successful the categories were encoded by ‘get_dummies()’.

Compiling, Training, and Evaluating the Model

Originally, I used 3 layers, 2 of which had 10 and 20 nodes in said layers, and the output layer only had one. The two hidden layers used ‘relu’ activation while the output used ‘sigmoid’ and we had a total of 200 epochs.

```
[ ]  # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
     input_features = len(X_train_scaled[0])
     hidden_nodes1=10
     hidden_nodes2=20

     nn = tf.keras.models.Sequential()

     # First hidden layer
     nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

     # Second hidden layer
     nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='relu'))

     # Output layer
     nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

     # Check the structure of the model
     nn.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_37 (Dense)            (None, 10)                440

 dense_38 (Dense)            (None, 20)                220

 dense_39 (Dense)            (None, 1)                 21

=================================================================
Total params: 681
Trainable params: 681
Non-trainable params: 0
_________________________________________________________________
```

As you can see, our accuracy resulted in less than our goal of 75%. As such, changes needed to be made, and it went through several iterations. This included changing the number of layers, neurons, their activation functions and even some of the data we used.

```
[ ]  # 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: 0.7291 - 361ms/epoch - 1ms/step
     Loss: 0.5547857284545898, Accuracy: 0.7290962338447571
```

In our final iteration, we used a total of 4 layers, the first 3 of which had 7, 14, and 21 nodes and only the first layer used activation function ‘relu’ while the rest used ‘sigmoid’. There was also a drop in epochs to 50. The major data change is that we actually KEPT the ‘NAME’ column in the data set and binned names that were present less than 100 times in the data. The idea being that we could more accurately predict repeat users/charities who used Alphabet Soup for funding.

```
+ Code  + Text

[ ]  # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
     # ATTEMPT 4, even LESS accurate? Keeping the name column but configuring the data to only be charities with 100 or more appearences, small nodes and epochs
     input_features = len(X_train_scaled[0])
     hidden_nodes1=7
     hidden_nodes2=14
     hidden_nodes3=21

     nn = tf.keras.models.Sequential()

     # First hidden layer
     nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

     # Second hidden layer
     nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='sigmoid'))

     # Third hidden layer
     nn.add(tf.keras.layers.Dense(units=hidden_nodes3, activation='sigmoid'))

     # 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_4 (Dense)             (None, 7)                 525

      dense_5 (Dense)             (None, 14)                112

      dense_6 (Dense)             (None, 21)                315

      dense_7 (Dense)             (None, 1)                 22

     =================================================================
     Total params: 974
     Trainable params: 974
     Non-trainable params: 0
     _________________________________________________________________
```

As you can see, the resulting accuracy results are 76%, 1% over our target!

```
268/268 - 0s - loss: 0.4925 - accuracy: 0.7602 - 499ms/epoch - 2ms/step
Loss: 0.4925134479999542, Accuracy: 0.7602332234382629
```

Summary:
We were able to meet our targets thanks to the changes made throughout the assignment. There's many different ways to experiment with neural networking to attempt to increase accuracy. You might be able to repeat this with a different model with more epochs, or less layers. A lot of it is experimentation for what works depending on each data set. But most importantly is to understand the data you have at your disposal. Without realizing how the 'NAME' data could be used, I may not have been able to reach the target.