Alphabet Soup Classification Model Report

Liberty Heise

Overview

The analysis was created to understand more about which factors would improve the success of particular charity organization over others. The data contained a wide variety of charity organizations with many different attributes which were then narrowed down to reflect which charity organizations would succeed and which would not.

Results

- **Data Preprocessing**

The target for this model is: Is_Successful.

The features for this model are: Application_Type, Affiliation, Classification, Use_Case, Organization, Status, Income_Amt, Special_Considerations, Ask_Amt.

The variables removed (neither targets or features) are: EIN and Name.

- Compiling, Training, and Evaluating the Model (Round One)

How many neurons, layers, and activation functions did you select for your neural network model, and why?

Were you able to achieve the target model performance? The target was .75, here is what I achived `Loss: 0.5988993048667908, Accuracy: 0.6997084617614746`

What steps did you take in your attempts to increase model performance?

I then created a second model (called Charity_Model_Two) and began to change the following in order to get a better response.

My first decision was to change the cutoff for both App_Type and and Class_Count. Initially, I set it at 100 for both and then changed it on my second model to 1000 and also 200.

```
APP_TYPE_COUNT_CUTOFF = 100
app_type_below_cutoff_s = at_vc_s < APP_TYPE_COUNT_CUTOFF
#print(app_type_below_cutoff_s)

# use the variable name `application_types_to_replace`
application_types_to_replace = at_vc_s[app_type_below_cutoff_s<100].index

# Replace in dataframe
for app in application_types_to_replace:
    clean_df['APPLICATION_TYPE'] = clean_df['APPLICATION_TYPE'].replace(app,"Other")

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

I applied changes to the number of epochs from 100 to 200 with an outcome that was even less accurate. I also changed the split to .25 from .15 with an outcome that was also worst, 0.697, than the original setup which stood at 0.699.

```
number_input_features = len( X_train_scaled[0])
hidden_node_1 = 3
hidden_node_2 = 4
hidden_node_3 = 5
nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node_1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_node_2, activation='relu'))

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

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 3)                 93

 dense_1 (Dense)             (None, 4)                 16

 dense_2 (Dense)             (None, 1)                 5
```

In addition to changing the epochs and split, I added more hidden nodes for a maximum of 4 and also changed the connections to higher numbers. I still was not able to render accuracy higher than .69 and removed the 4th note and lowered the numbers.

Summary

Overall, the model used to produce loss and accuracy was decent but not high enough of an accuracy to be useful in prediction. In order to create a more accurate model, a review of how the data features iinteract with each other is crucial. Once a deeper study of the data has been made, a few models could be more useful, such as decision trees, random forests or logistic regression.