Charity Predictor Analysis

Overview:

Alphabet Soup is a nonprofit foundation looking for a tool for selecting applicants for funding. The tool's purpose is to create an increased chance of better-funded ventures. I set myself with the goal of having 75% accuracy in predicting the chance of a venture being successful under Alphabet Soup. Alphabet Soup sent me a CSV with data from 34,000 organizations that they have funded. I used this CSV as I created a neural network to create the tool.

Results:

Data Preprocessing

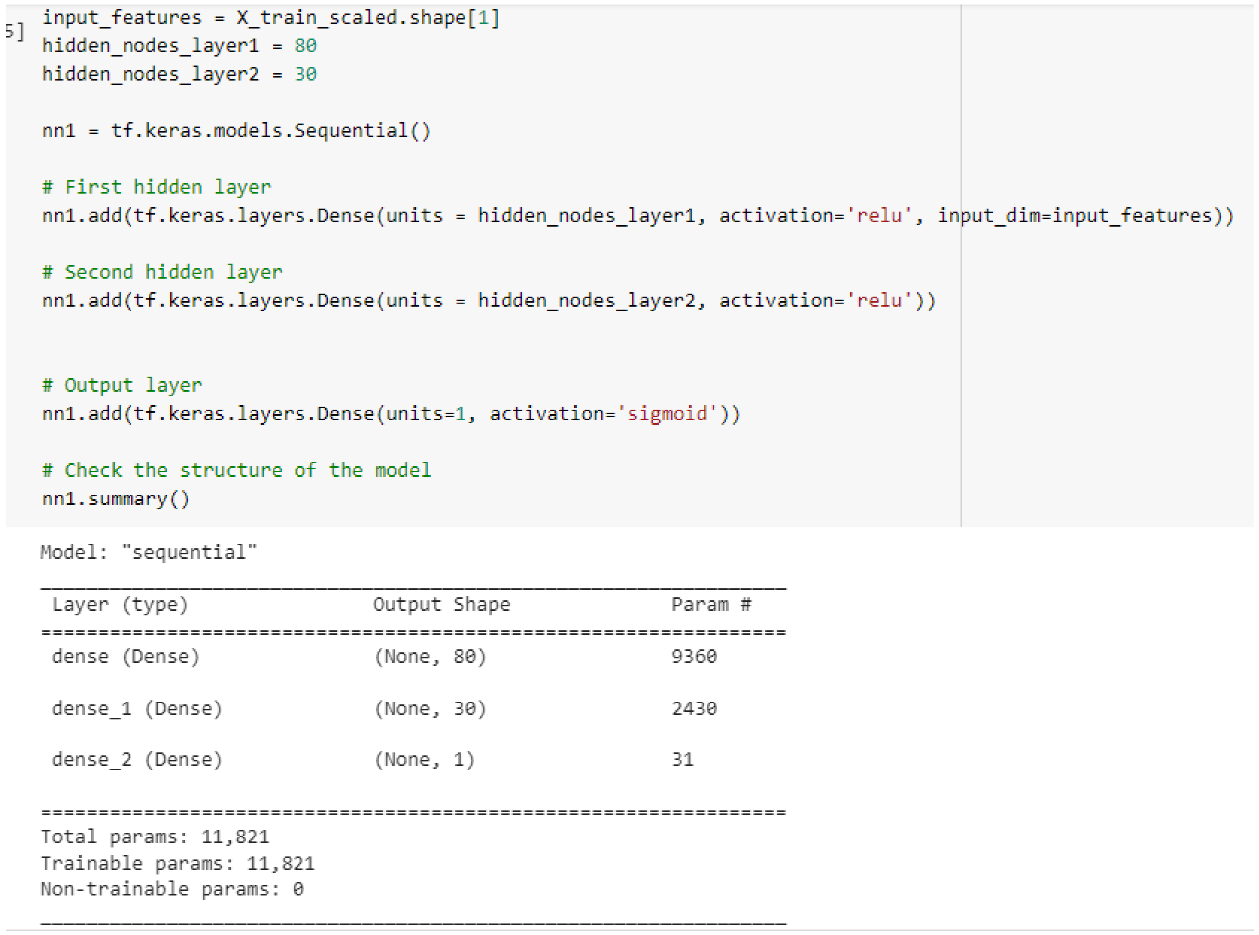
The target for my model was the column “Is_Successful”. I chose this as my target variable because it is the output I am looking to predict with my model, is the organization successful or not.

The features for my model are the columns, "Application_Type, Affiliation, Classification, Use_Case, Organization, Status, Income_Amt, Special_Considerations, and Ask_AMT.” All of these columns play some part in predicting if an organization will be successful.

I got rid of the columns, “EIN, and Name”. These columns were not going to have an effect on the outcome of an organization were just creating noise so it is best to remove them.

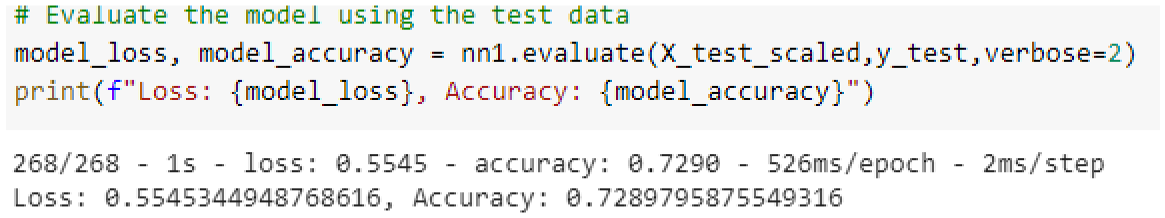
Compiling, Training, and Evaluating the Model

I created three models to attempt to reach my goal of 75%. In my first model I used two hidden layers, the first had 80 neurons and the second had 30 neurons. I chose to use “relu” as my activation function and 50 EPOCHS. This gave me 72.89%.



```python
input_features = X_train_scaled.shape[1]
hidden_nodes_layer1 = 80
hidden_nodes_layer2 = 30

nn1 = tf.keras.models.Sequential()

# First hidden layer
nn1.add(tf.keras.layers.Dense(units = hidden_nodes_layer1, activation='relu', input_dim=input_features))

# Second hidden layer
nn1.add(tf.keras.layers.Dense(units = hidden_nodes_layer2, activation='relu'))


# Output layer
nn1.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn1.summary()
```

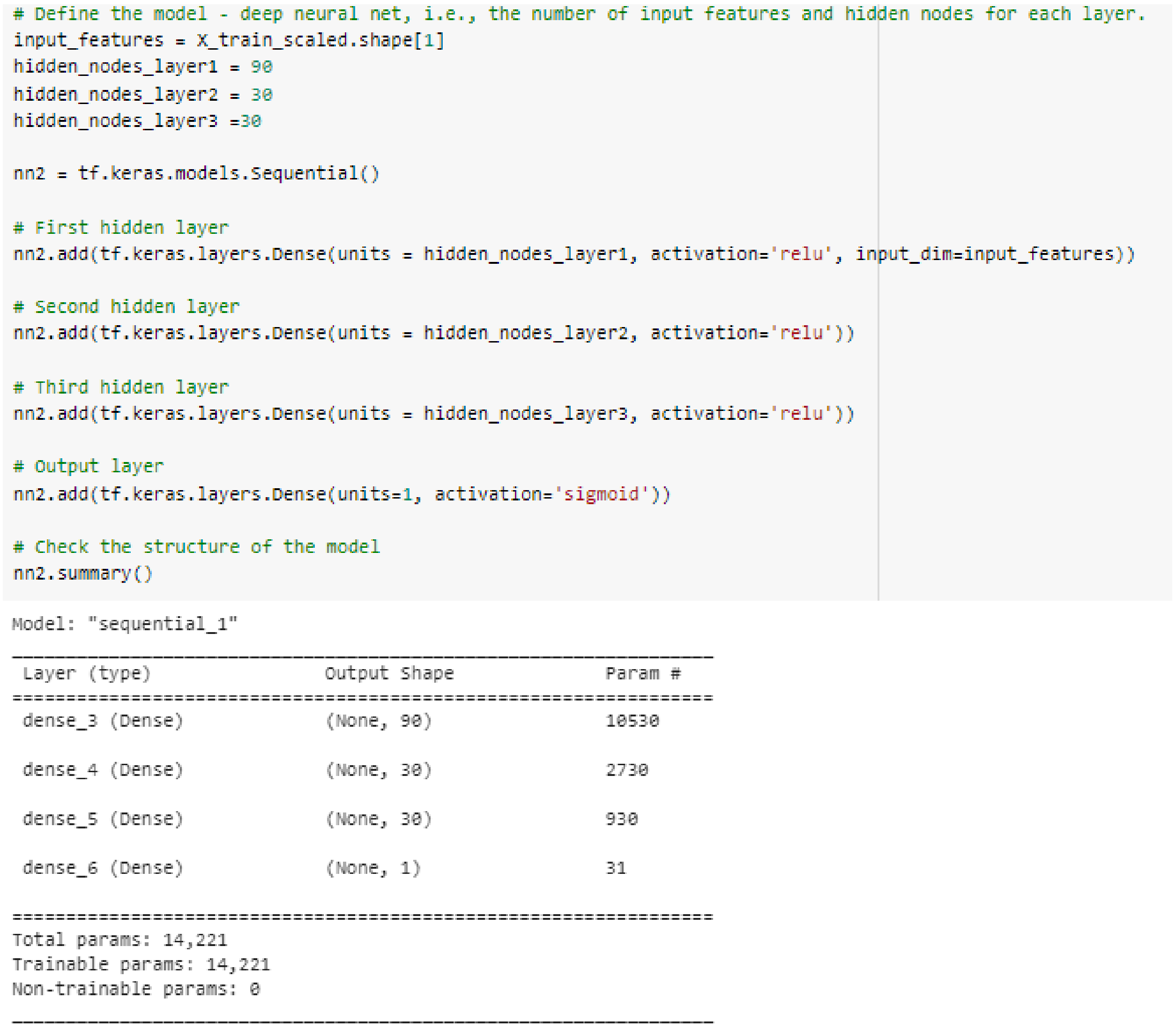
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                9360

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 11,821
Trainable params: 11,821
Non-trainable params: 0
_________________________________________________________________
```



```python
# Evaluate the model using the test data
model_loss, model_accuracy = nn1.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

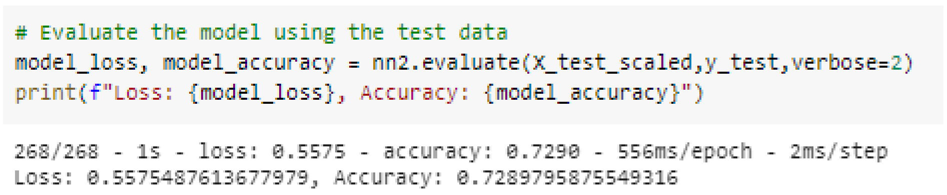
```
268/268 - 1s - loss: 0.5545 - accuracy: 0.7290 - 526ms/epoch - 2ms/step
Loss: 0.5545344948768616, Accuracy: 0.7289795875549316
```

In my second model, I added a third hidden layer, the first had 90 neurons, the second had 30 neurons, and the third had 30 neurons. I kept “relu” as my activation function and 50 EPOCHS. This did not change my percentage as I was still at 72.89%.



```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_features = X_train_scaled.shape[1]
hidden_nodes_layer1 = 90
hidden_nodes_layer2 = 30
hidden_nodes_layer3 =30

nn2 = tf.keras.models.Sequential()

# First hidden layer
nn2.add(tf.keras.layers.Dense(units = hidden_nodes_layer1, activation='relu', input_dim=input_features))

# Second hidden layer
nn2.add(tf.keras.layers.Dense(units = hidden_nodes_layer2, activation='relu'))

# Third hidden layer
nn2.add(tf.keras.layers.Dense(units = hidden_nodes_layer3, activation='relu'))

# Output layer
nn2.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn2.summary()
```

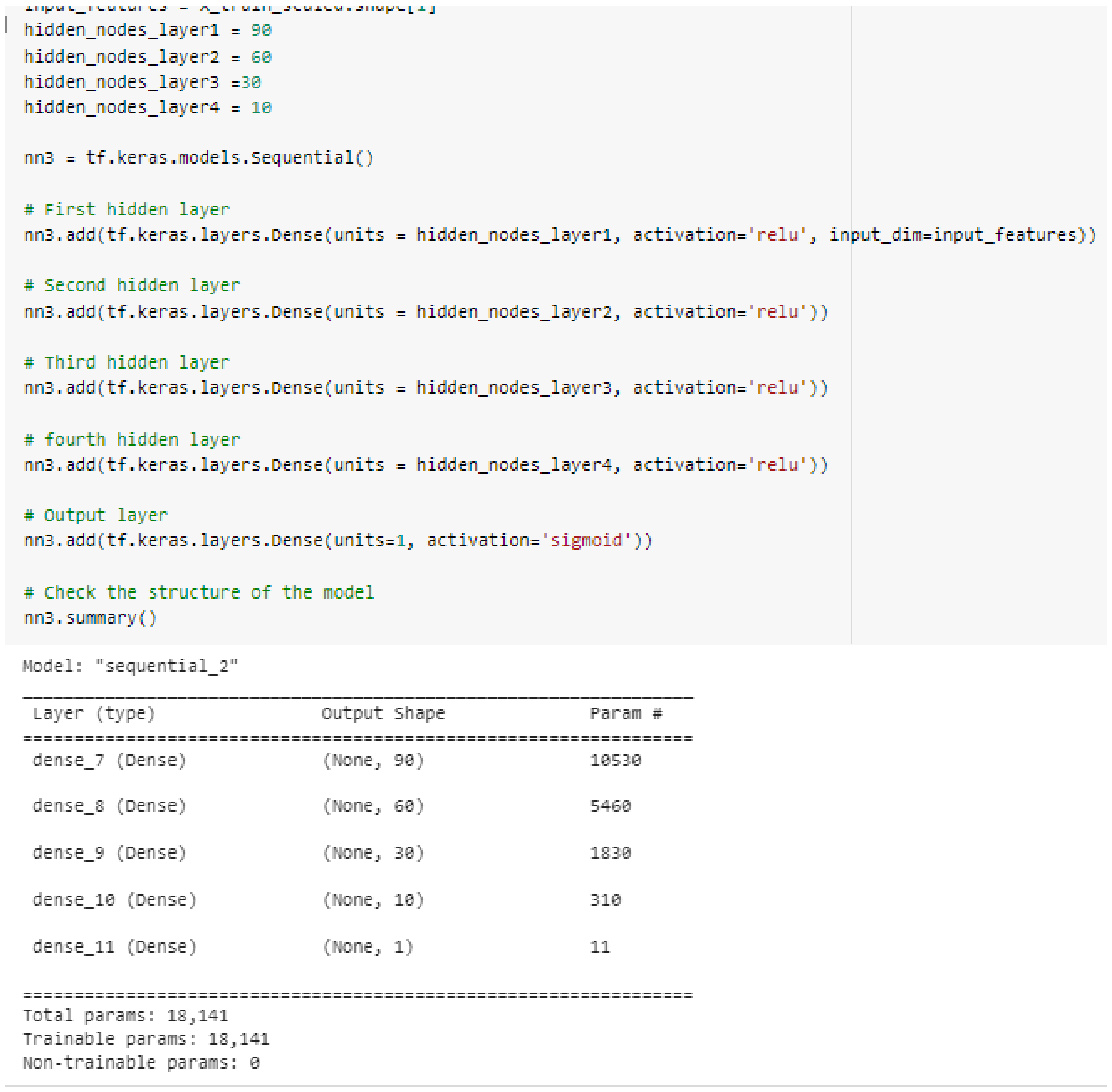
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 90)                10530

 dense_4 (Dense)             (None, 30)                2730

 dense_5 (Dense)             (None, 30)                930

 dense_6 (Dense)             (None, 1)                 31

=================================================================
Total params: 14,221
Trainable params: 14,221
Non-trainable params: 0
_________________________________________________________________
```



```
# Evaluate the model using the test data
model_loss, model_accuracy = nn2.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

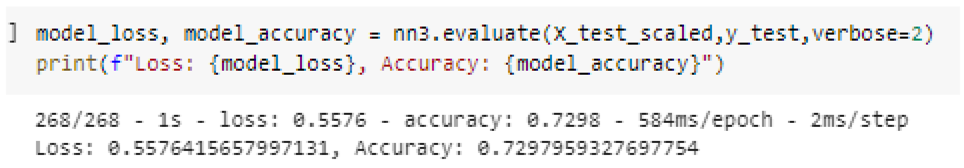
```
268/268 - 1s - loss: 0.5575 - accuracy: 0.7290 - 556ms/epoch - 2ms/step
Loss: 0.5575487613677979, Accuracy: 0.7289795875549316
```

In my third model, I added a fourth hidden layer, the first had 90 neurons, the second had 60 neurons, the third had 30 neurons, and the fourth had 10 neurons. I once again kept “relu” as my activation function and used 50 EPOCHS. Once again I had only slight improvement with a 72.97%.



```
input_features = X_train_scaled.shape[1]
hidden_nodes_layer1 = 90
hidden_nodes_layer2 = 60
hidden_nodes_layer3 =30
hidden_nodes_layer4 = 10

nn3 = tf.keras.models.Sequential()

# First hidden layer
nn3.add(tf.keras.layers.Dense(units = hidden_nodes_layer1, activation='relu', input_dim=input_features))

# Second hidden layer
nn3.add(tf.keras.layers.Dense(units = hidden_nodes_layer2, activation='relu'))

# Third hidden layer
nn3.add(tf.keras.layers.Dense(units = hidden_nodes_layer3, activation='relu'))

# fourth hidden layer
nn3.add(tf.keras.layers.Dense(units = hidden_nodes_layer4, activation='relu'))

# Output layer
nn3.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_7 (Dense)             (None, 90)                10530     
                                                                 
 dense_8 (Dense)             (None, 60)                5460      
                                                                 
 dense_9 (Dense)             (None, 30)                1830      
                                                                 
 dense_10 (Dense)            (None, 10)                310       
                                                                 
 dense_11 (Dense)            (None, 1)                 11        
                                                                 
=================================================================
Total params: 18,141
Trainable params: 18,141
Non-trainable params: 0
```



```
] model_loss, model_accuracy = nn3.evaluate(X_test_scaled,y_test,verbose=2)
  print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")

  268/268 - 1s - loss: 0.5576 - accuracy: 0.7298 - 584ms/epoch - 2ms/step
  Loss: 0.5576415657997131, Accuracy: 0.7297959327697754
```

Summary:

After all three models, I was only able to reach 72.8% accuracy, which is lower than the goal of 75%. It is possible to increase the accuracy by changing some other features, how many neurons are in a hidden layer, the number of hidden layers, the activation function used, etc... From my experience adjusting these factors only played a minor factor in the accuracy of the model. It is because of this I would recommend using a different model to predict whether it would be a successful venture for Alphabet Soup to fund an organization.