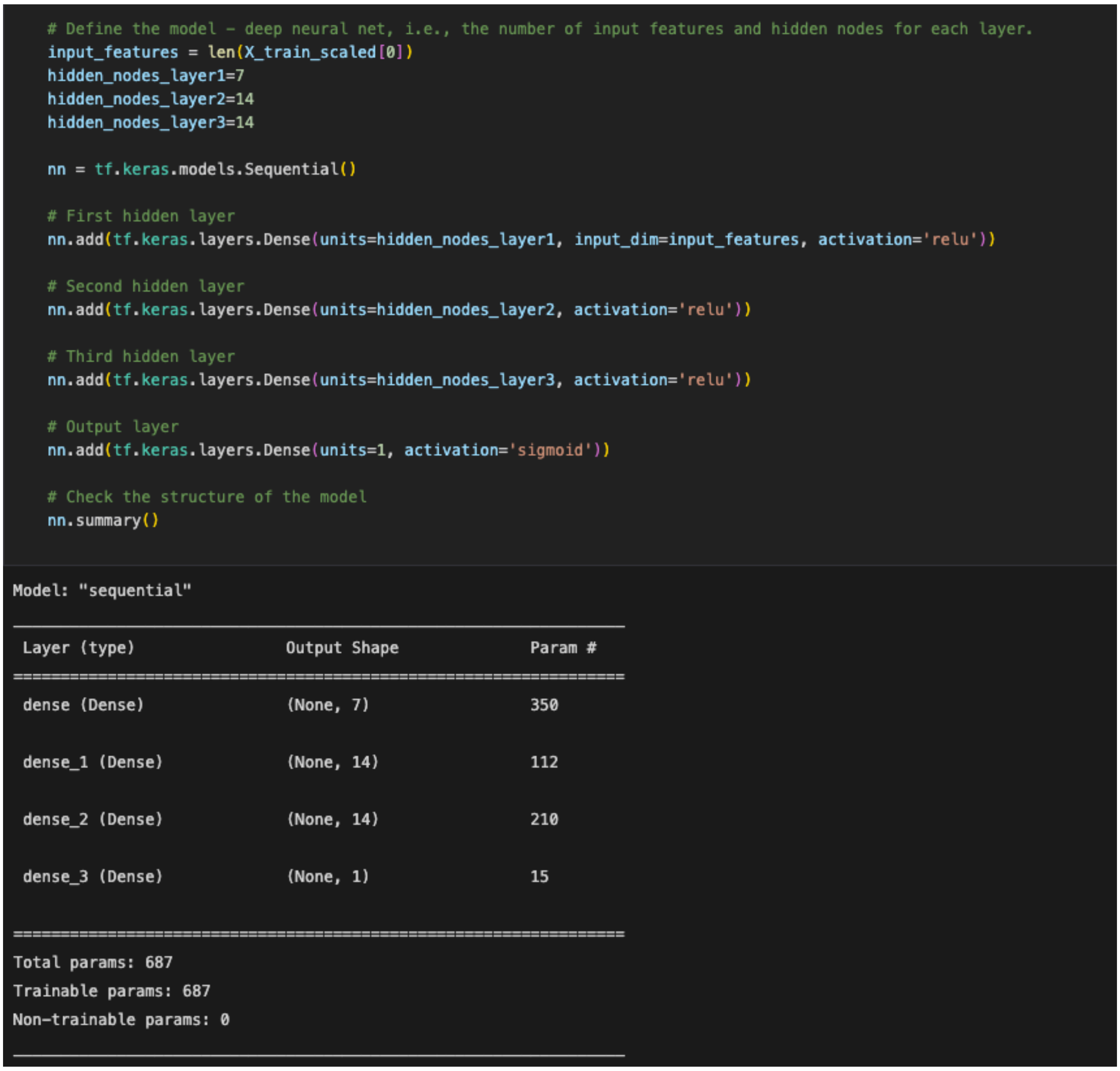
Deep Learning Challenge: Charity Funding Predictor

1. **Overview:** A non-profit foundation named "Alphabet Soup" is looking to create an algorithm to predict whether applicants applying for funding will be successful or not. We are to use our knowledge of machine learning to create a classifier able to successfully predict the outcome of applicants applying for funding.

2. Results:

- **Data Preprocessing:**
  - Target was the column named "IS_SUCCESSFUL".
  - Features of the model were categorically created using "get_dummies".
  - Columns "EIN" and "NAME" were dropped as they don't add value to the analysis.

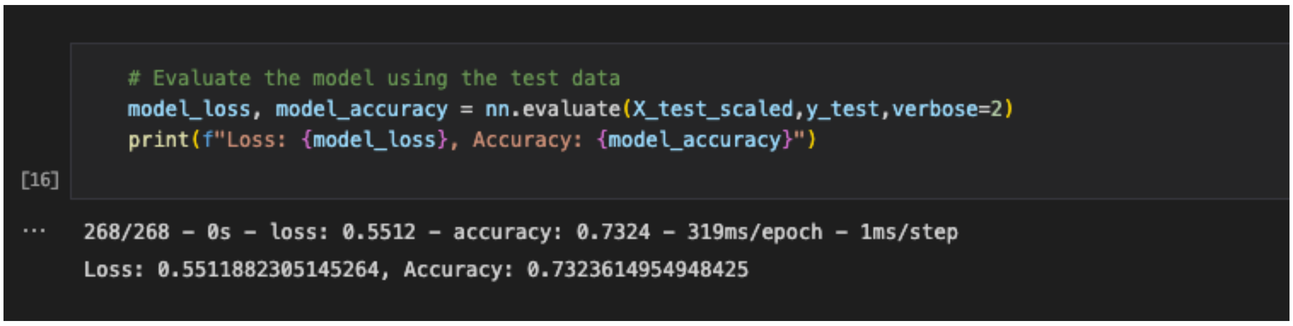
- **Compiling, Training, and Evaluating the Model:**
  - Number of neurons range between 21 and 45 in the different trials created to increase the accuracy. Number or layers are either 2 or 3 layers (excluding the output layer). The activation function is "sigmoid" for all of the trials created. A screenshot of the first trial is shown below:



```
# 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_nodes_layer1=7
hidden_nodes_layer2=14
hidden_nodes_layer3=14

nn = tf.keras.models.Sequential()

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

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

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, 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, 7)                 350

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 14)                210

 dense_3 (Dense)             (None, 1)                 15

=================================================================
Total params: 687
Trainable params: 687
Non-trainable params: 0
_________________________________________________________________
```

- I was NOT able to achieve the required performance of 75% accuracy. The highest accuracy achieved was 73.2% as shown below:



```
# 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}")
```

[16]

```
268/268 - 0s - loss: 0.5512 - accuracy: 0.7324 - 319ms/epoch - 1ms/step
Loss: 0.5511882305145264, Accuracy: 0.7323614954948425
```

- I took the following steps to attempt to increase the model's performance:
A. Add another hidden layer.
B. Increase number of nodes.
C. Increase and decrease epochs.
D. Increase number of values in bins.

3. Summary:
I ran the model once before I had 3 optimization runs to try to get the accuracy above 75%. I added more layers, more nodes, changed the number of epochs, and increased the values in each bin.

I am not sure what steps I can take to increase the accuracy. Changing the activation function might help, so I would recommend doing that to try to enhance the accuracy.