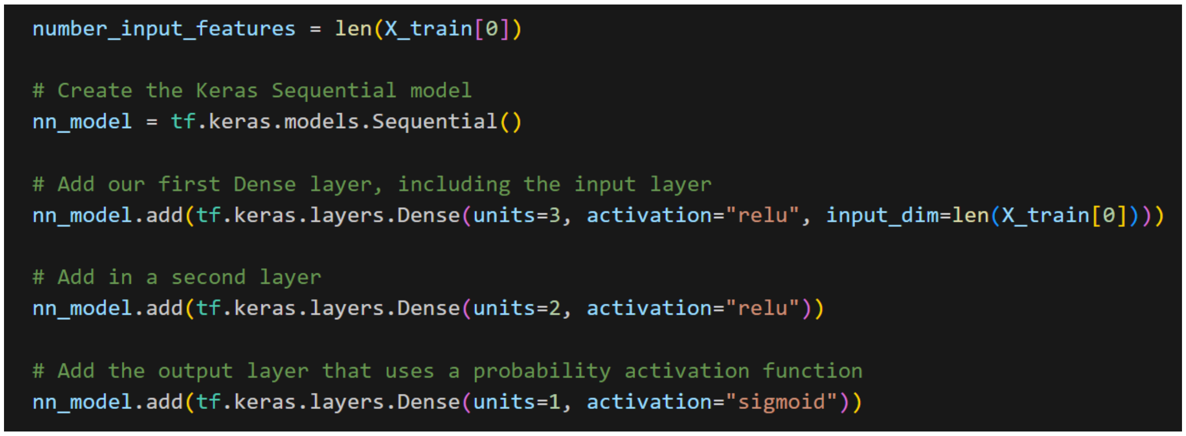
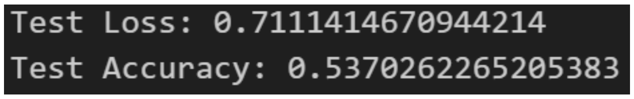
Module 22 Challenge Report

The purpose of this analysis was to design a tool that can help the nonprofit foundation Alphabet Soup select applicants for funding with the best chance of success in their ventures. We used neural networks and machine learning to create a binary classifier that can predict whether applicants will be successful in funded by Alphabet Soup. The target of our model was to predict whether our applicants would be successful or not. Our dependent variable was a binary classifier column named IS_SUCCESSFUL. The features for our model were the classification, application type, and ask amount. We removed the columns 'EIN' and 'NAME' because they are neither targets nor features.

We started with two layers. They had three and two neurons respectively and both used the relu activation function. There was an output layer with one neuron and a sigmoid activation function. Our model performance did not achieve the desired result with this setup:



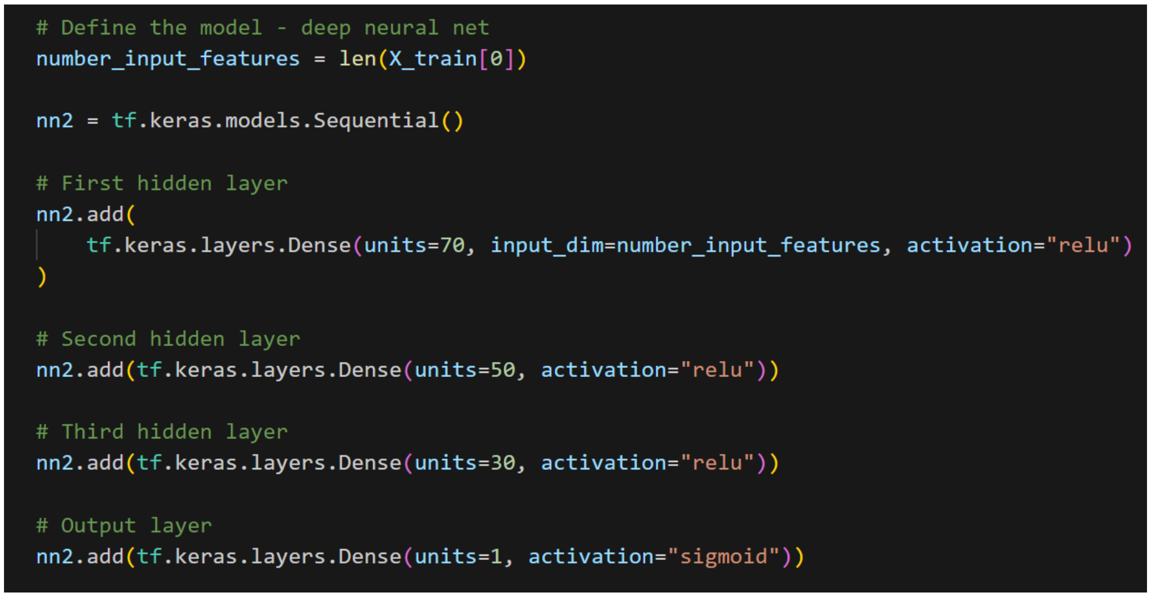
```
number_input_features = len(X_train[0])

# Create the Keras Sequential model
nn_model = tf.keras.models.Sequential()

# Add our first Dense layer, including the input layer
nn_model.add(tf.keras.layers.Dense(units=3, activation="relu", input_dim=len(X_train[0])))

# Add in a second layer
nn_model.add(tf.keras.layers.Dense(units=2, activation="relu"))

# Add the output layer that uses a probability activation function
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))
```

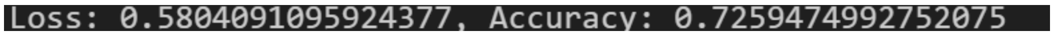


```
Test Loss: 0.7111414670944214
Test Accuracy: 0.5370262265205383
```

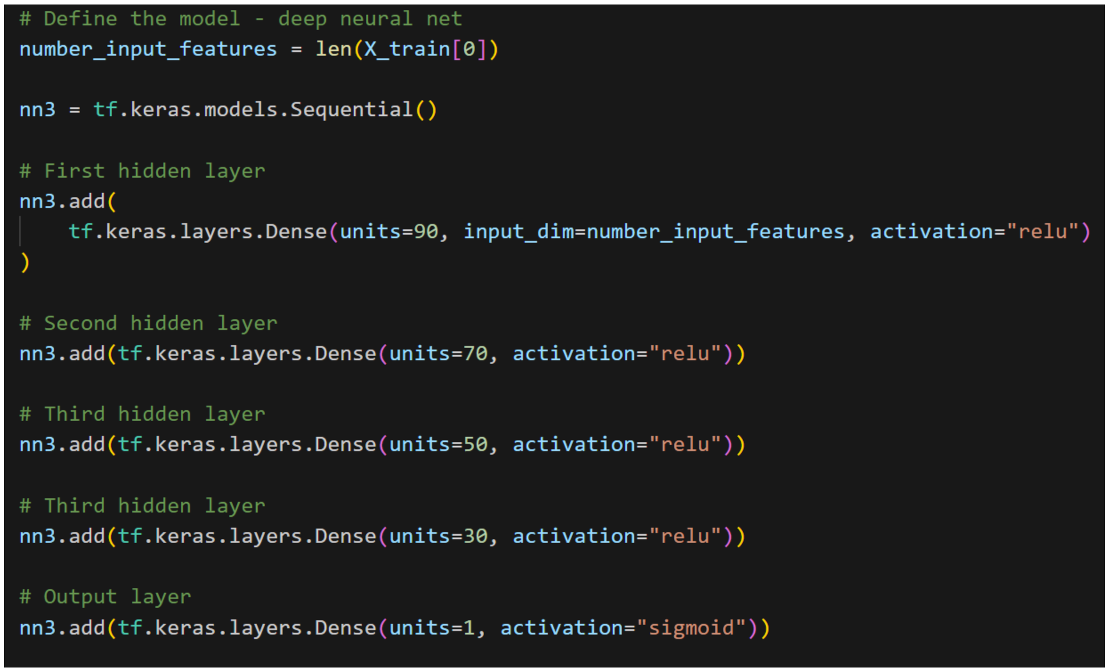
After evaluating our first model two more models were created. In each we increased the number of layers and neurons in those layers. The first improvement found much higher results, then no improvement with a third.

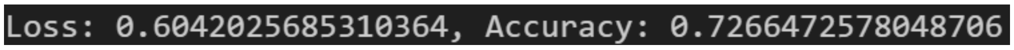
Model 2:


```
# Define the model - deep neural net
number_input_features = len(X_train[0])

nn2 = tf.keras.models.Sequential()

# First hidden layer
nn2.add(
    tf.keras.layers.Dense(units=70, input_dim=number_input_features, activation="relu")
)

# Second hidden layer
nn2.add(tf.keras.layers.Dense(units=50, activation="relu"))

# Third hidden layer
nn2.add(tf.keras.layers.Dense(units=30, activation="relu"))

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


```
Loss: 0.5804091095924377, Accuracy: 0.7259474992752075
```

Model 3:


```
# Define the model - deep neural net
number_input_features = len(X_train[0])

nn3 = tf.keras.models.Sequential()

# First hidden layer
nn3.add(
    tf.keras.layers.Dense(units=90, input_dim=number_input_features, activation="relu")
)

# Second hidden layer
nn3.add(tf.keras.layers.Dense(units=70, activation="relu"))

# Third hidden layer
nn3.add(tf.keras.layers.Dense(units=50, activation="relu"))

# Third hidden layer
nn3.add(tf.keras.layers.Dense(units=30, activation="relu"))

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


```
Loss: 0.6042025685310364, Accuracy: 0.7266472578048706
```

The deep learning model we created was able to achieve a peak accuracy of 72%. With further development we think this could be improved upwards of 75%. This was a Keras model but we believe an XGB model would also perform quite well in this classification problem due to it being equally capable of taking on classification problems.