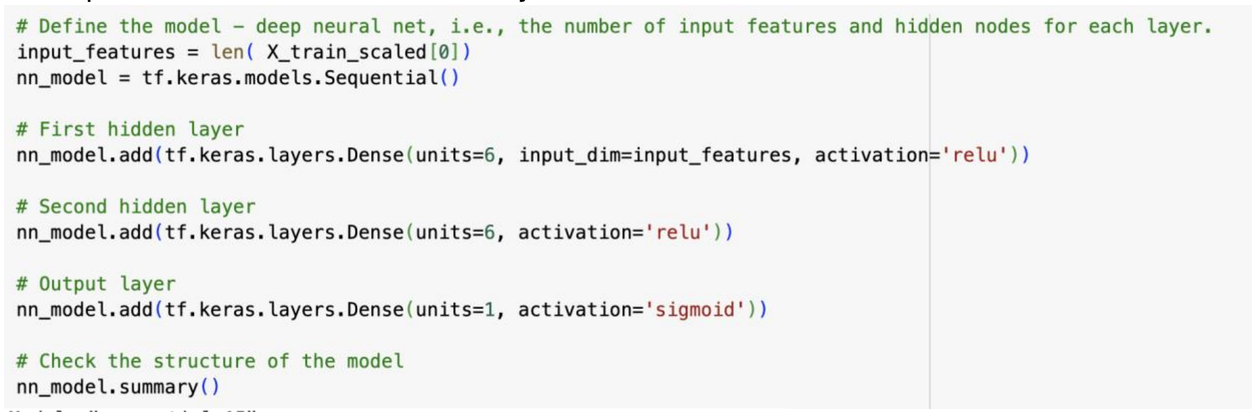
Overview:
The purpose of this analysis was to develop a model that could best predict applicants for the Alphabet Soup funding that would have the greatest success in their future ventures. We used a dataset that contained over 34,000 organizations that received funding from Alphabet Soup and contained information about the organizations' income amount, government classification, status, organization type, affiliation, application type, and other information.

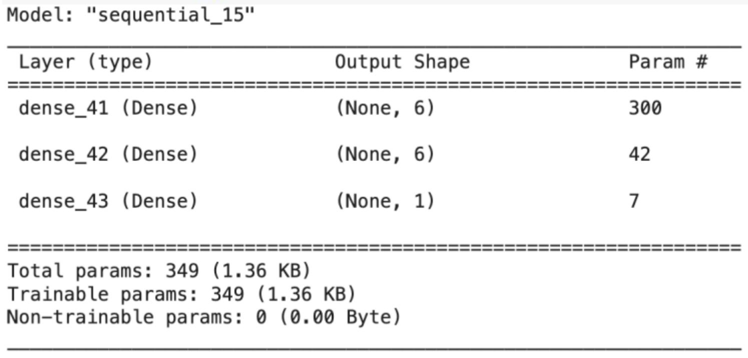
Results:
We generated models based off the organization's application type to determine if they would be successful or not.

The target for the model is "IS_SUCCESSFUL" which codes whether or not the organization was successful (1=successful, 0=not successful). We analyzed the "APPLICATION" column and used "CLASSIFICATION" column for binning. We removed the "NAME" and "EIN" columns from the data set as they were not considered relevant targets or features.

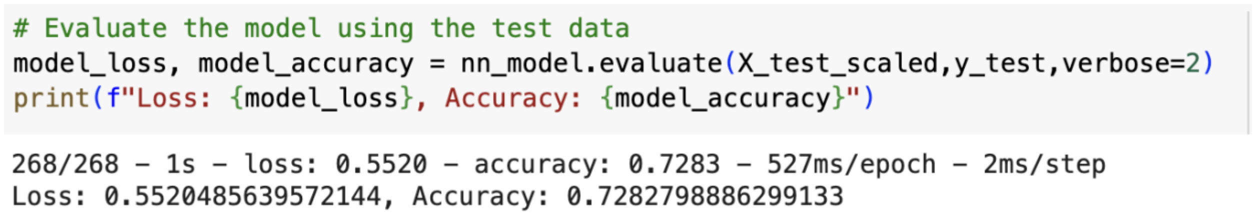
The first model contained three layers. The first two layers had 6 neurons and relu activation shapes, and the last layer had 1 neuron and a sigmoid activation shape. The first attempt resulted in a 72.83% accuracy.



```
# 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])
nn_model = tf.keras.models.Sequential()

# First hidden layer
nn_model.add(tf.keras.layers.Dense(units=6, input_dim=input_features, activation='relu'))

# Second hidden layer
nn_model.add(tf.keras.layers.Dense(units=6, activation='relu'))

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

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



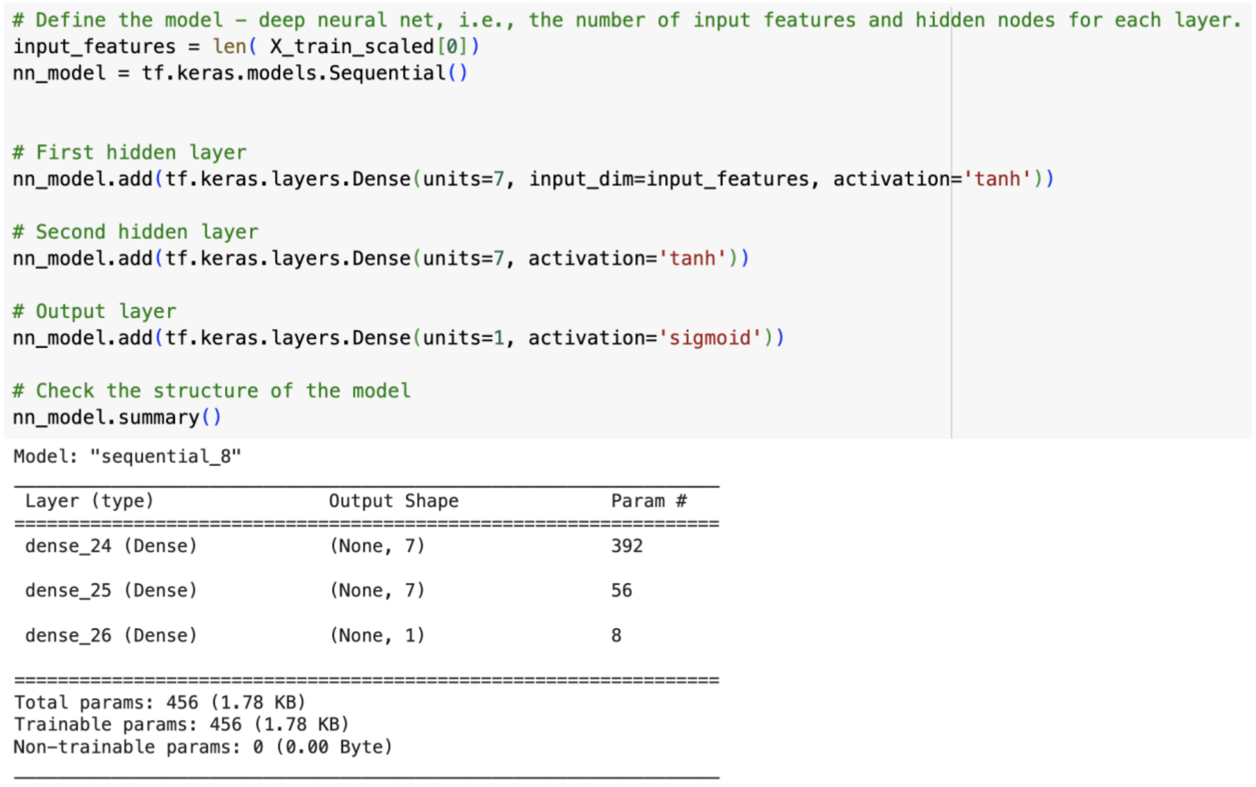
```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_41 (Dense)            (None, 6)                 300

 dense_42 (Dense)            (None, 6)                 42

 dense_43 (Dense)            (None, 1)                 7

=================================================================
Total params: 349 (1.36 KB)
Trainable params: 349 (1.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



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

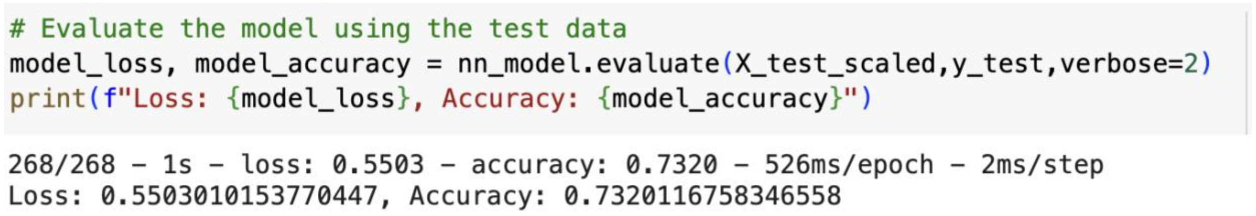
```
268/268 - 1s - loss: 0.5520 - accuracy: 0.7283 - 527ms/epoch - 2ms/step
Loss: 0.5520485639572144, Accuracy: 0.7282798886299133
```

The second model also contained three layers. The first two layers had 7 neurons with tanh activation shapes and the last layer had 1 neuron with a sigmoid activation shape. However, it analyzed the “FUNDING_AMT” variable and used the “CLASSIFICATION” column for binning. It’s resulting accuracy was 73.2%. Unfortunately, after a few iterations, I was able to increase the accuracy but wasn’t able to achieve the target model performance. I changed the number of neurons in each layer, the features that were examined, and the activation shape, but could only yield slightly higher model performance.



```
# 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])
nn_model = tf.keras.models.Sequential()


# First hidden layer
nn_model.add(tf.keras.layers.Dense(units=7, input_dim=input_features, activation='tanh'))

# Second hidden layer
nn_model.add(tf.keras.layers.Dense(units=7, activation='tanh'))

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

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

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_24 (Dense)            (None, 7)                 392       
                                                                 
 dense_25 (Dense)            (None, 7)                 56        
                                                                 
 dense_26 (Dense)            (None, 1)                 8         
                                                                 
=================================================================
Total params: 456 (1.78 KB)
Trainable params: 456 (1.78 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



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

```
268/268 - 1s - loss: 0.5503 - accuracy: 0.7320 - 526ms/epoch - 2ms/step
Loss: 0.5503010153770447, Accuracy: 0.7320116758346558
```

Summary:
Overall, the model performed just shy of its 75% target performance. Further analysis should be done to see if other activation shapes, other features, or more hidden layers would be able to model the dataset better. Given the accuracy of the above two models, I would recommend the second one. It seems that looking at amount of funding compared to the application type is a better predictor of the success of a company.