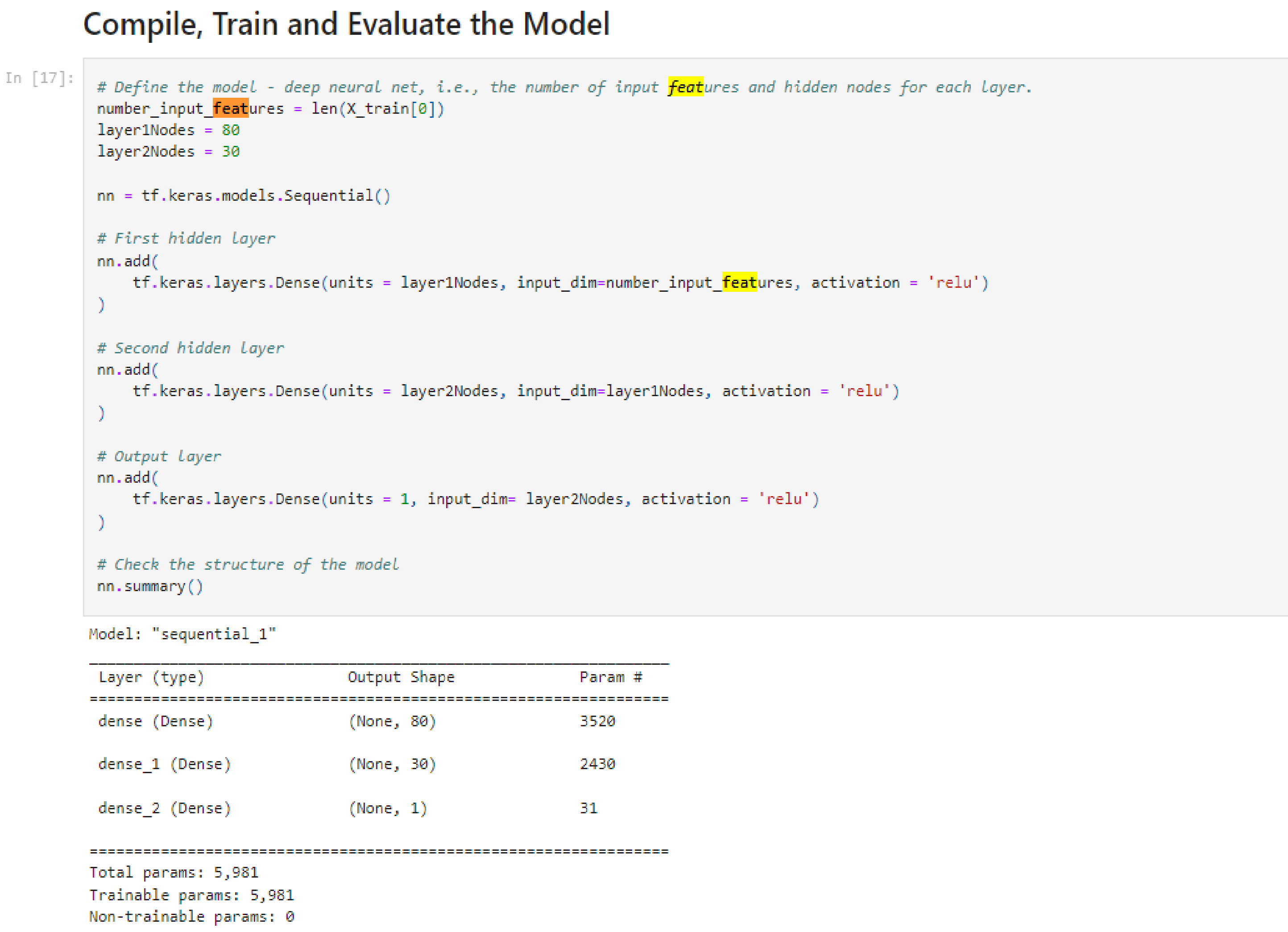
Step 4: Write a Report on the Neural Network Model

The purpose of this analysis was to analyze the data from a charity, Alphabet Soup, and use an algorithm to predict whether or not applicants for funding would be successful.

Data Preprocessing:

- What variable(s) are considered the target(s) for your model?
    - The 'is_successful' column and is represented by a 1 for a successful charity and a 0 for an unsuccessful charity.
- What variable(s) are considered to be the features for your model?
    - The 'Application_Type' column as well as the 'Classification' column. These columns were significant because they had over 10 unique values
- What variable(s) are neither targets nor features, and should be removed from the input data?
    - The 'EIN' and 'NAME' were dropped from the model, the rest of the columns were kept in.



Compile, Train and Evaluate the Model

In [17]:

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train[0])
layer1Nodes = 80
layer2Nodes = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(
    tf.keras.layers.Dense(units = layer1Nodes, input_dim=number_input_features, activation = 'relu')
)

# Second hidden layer
nn.add(
    tf.keras.layers.Dense(units = layer2Nodes, input_dim=layer1Nodes, activation = 'relu')
)

# Output layer
nn.add(
    tf.keras.layers.Dense(units = 1, input_dim= layer2Nodes, activation = 'relu')
)

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

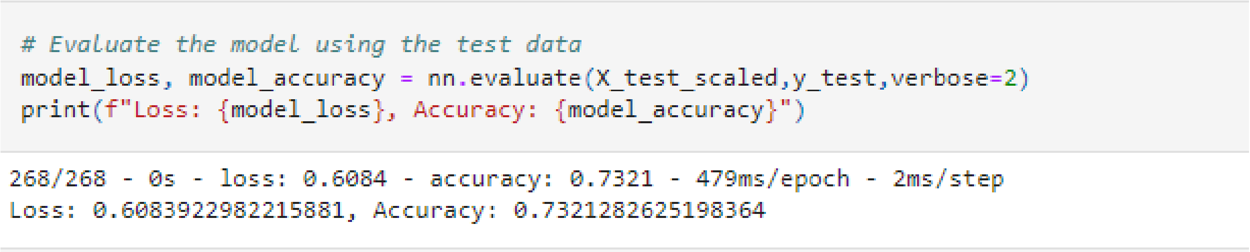
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                3520

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 5,981
Trainable params: 5,981
Non-trainable params: 0
```

Compiling, Training, and Evaluating the Model

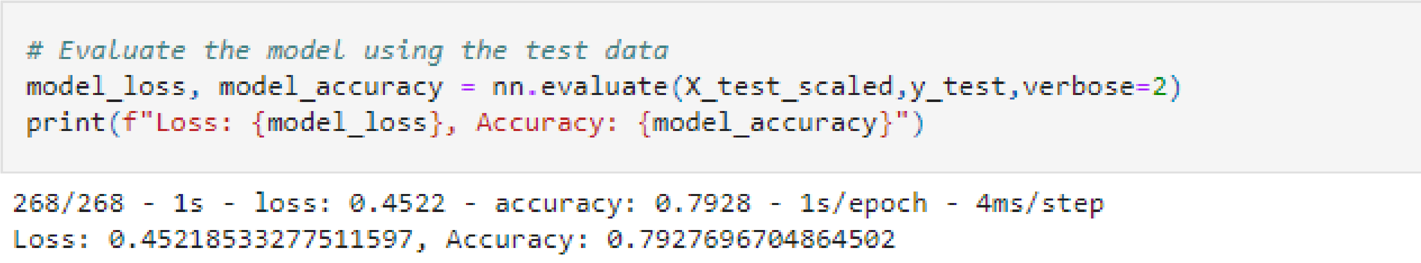


```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.6084 - accuracy: 0.7321 - 479ms/epoch - 2ms/step
Loss: 0.6083922982215881, Accuracy: 0.7321282625198364
```

- How many neurons, layers, and activation functions did you select for your neural network model, and why?
  - Neural networks were applied for each model on three layers each. The number of features dictated the number of hidden nodes
- Were you able to achieve the target model performance?
  - A three-layer model generated 5,981 parameters and 73% accuracy, lower than the requested 75%
- What steps did you take to try and increase model performance?
  - The optimization added ‘NAME’ back into the dataset and the accuracy score was 79%, an increase of 6%

Multiple layers are useful for deep learning models, it increases accuracy for the machine in predicting and classifying information..



```
# 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}")
```

```
268/268 - 1s - loss: 0.4522 - accuracy: 0.7928 - 1s/epoch - 4ms/step
Loss: 0.45218533277511597, Accuracy: 0.7927696704864502
```