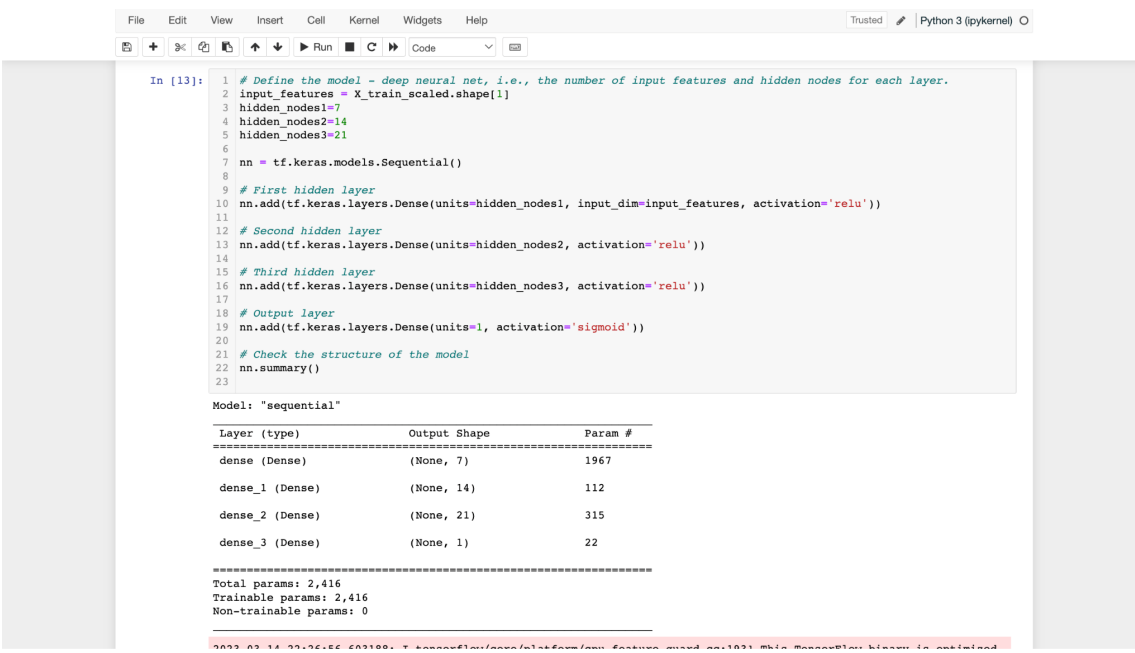
Overview:
The nonprofit foundation Alphabet Soup wanted a tool to help them select applicants for funding. They need a binary classifier to predict whether applicants will be successful if funded by The Alphabet Soup.
‘

Data Preprocessing:
Unnecessaries such as EIN and Name were removed from the dataset and all remaining metrics were considered in the model. Both Classification and Application type were features for the model.

- What variables are the target for your model?
- What variables are the features for your model?
- What variables should be removed from the input data because they are neither target nor features?

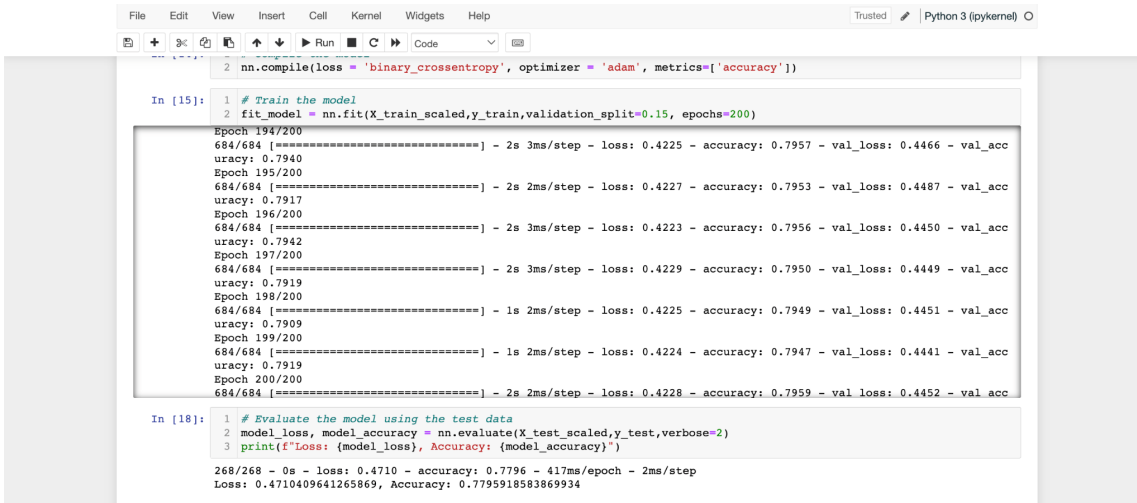
Compiling,Training,and Evaluating the Model:

Neural Network was used on each model and originally set with 2. For the final model, 3 layers were added that helped achieve an accuracy of 75%.



```
In [13]:
# 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_nodes1=7
hidden_nodes2=14
hidden_nodes3=21

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='relu'))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes3, 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)                 1967

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 21)                315

 dense_3 (Dense)             (None, 1)                 22

=================================================================
Total params: 2,416
Trainable params: 2,416
Non-trainable params: 0
_________________________________________________________________
```

In order to achieve the model performance, I kept Name in the model and applied Name as a feature and binned the values. I kept classification as a feature in the model as well. In addition to the changes previously mentioned, I also added a third layer and changed the epochs to 200 instead of 100.



```
nn.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'])
```

In [15]:

```
# Train the model
fit_model = nn.fit(X_train_scaled,y_train,validation_split=0.15, epochs=200)
```

```
Epoch 194/200
684/684 [==============================] - 2s 3ms/step - loss: 0.4225 - accuracy: 0.7957 - val_loss: 0.4466 - val_acc
uracy: 0.7940
Epoch 195/200
684/684 [==============================] - 2s 2ms/step - loss: 0.4227 - accuracy: 0.7953 - val_loss: 0.4487 - val_acc
uracy: 0.7917
Epoch 196/200
684/684 [==============================] - 2s 3ms/step - loss: 0.4223 - accuracy: 0.7956 - val_loss: 0.4450 - val_acc
uracy: 0.7942
Epoch 197/200
684/684 [==============================] - 2s 3ms/step - loss: 0.4229 - accuracy: 0.7950 - val_loss: 0.4449 - val_acc
uracy: 0.7919
Epoch 198/200
684/684 [==============================] - 1s 2ms/step - loss: 0.4225 - accuracy: 0.7949 - val_loss: 0.4451 - val_acc
uracy: 0.7909
Epoch 199/200
684/684 [==============================] - 1s 2ms/step - loss: 0.4224 - accuracy: 0.7947 - val_loss: 0.4441 - val_acc
uracy: 0.7919
Epoch 200/200
684/684 [==============================] - 2s 2ms/step - loss: 0.4228 - accuracy: 0.7959 - val_loss: 0.4452 - val_acc
```

In [18]:

```
# 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: 0.4710 - accuracy: 0.7796 - 417ms/epoch - 2ms/step
Loss: 0.471040964126586 9, Accuracy: 0.7795918583869934
```

Summary:

Several layers should be considered, so that it can continue to predict and classify information based on the model.