1. **Overview of the analysis**:

   The purpose of the project is to develop a machine learning model that acts as a binary classifier. Using the provided dataset and machine learning techniques, the goal is to use the model that can effectively predict whether an applicant will be successful if funded by Alphabet Soup.

2. **Results**:

   - Data Preprocessing
     - What variable(s) are the target(s) for your model?
       - IS_SUCCESSFUL
     - What variable(s) are the features for your model?
       - APPLICATION_TYPE
       - AFFILIATION
       - CLASSIFICATION
       - USE_CASE
       - ORGANIZATION
       - STATUS
       - INCOME_AMT
       - SPECIAL_CONSIDERATIONS
       - ASK_AMT
     - What variable(s) should be removed from the input data because they are neither targets nor features?
       - EIN and NAME
   - Compiling, Training, and Evaluating the Model
     - How many neurons, layers, and activation functions did you select for your neural network model, and why? (first model before optimizing)
       - Number of neurons: The number of neurons in the first hidden layer is 80 and the number of neurons in the second hidden layer is 30.
       - Number of layers: The model has two hidden layers.

- Activation functions: The activation function used in both hidden layers is the Rectified Linear Unit (ReLU) activation function. The sigmoid activation function is used for the output.
- Reason: I took the neurons numbers provided in the course materials to begin with a baseline model.

Before Optimizing

```
Model: "sequential"
_________________________________________________________________
 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
```

First Optimizing

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               10368

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 16)                528

 dense_4 (Dense)             (None, 1)                 17

=================================================================
Total params: 21,249
Trainable params: 21,249
Non-trainable params: 0
_________________________________________________________________
```

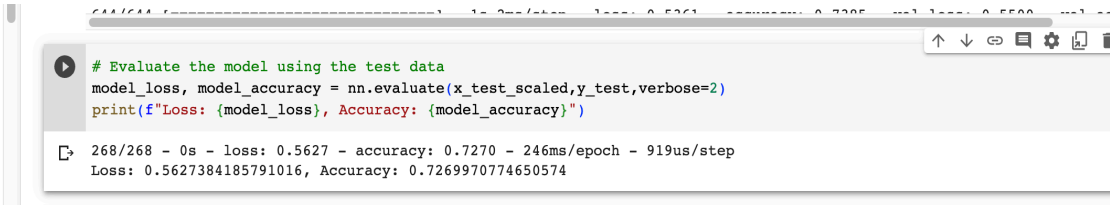
Second Optimizing

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 90)                40860

 dense_7 (Dense)             (None, 30)                2730

 dense_8 (Dense)             (None, 1)                 31

=================================================================
Total params: 43,621
Trainable params: 43,621
Non-trainable params: 0
_________________________________________________________________
```

- Were you able to achieve the target model performance?

I was not able to achieve the target model performance.

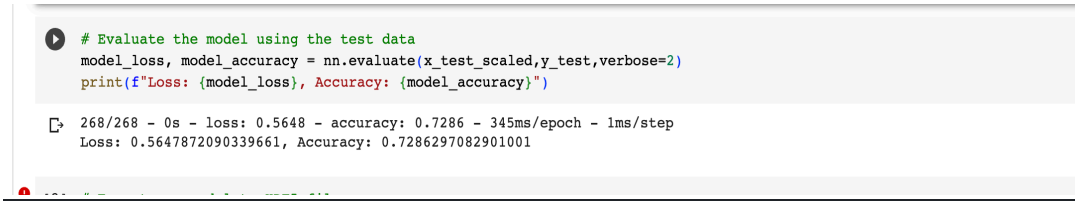
First try:



```
# 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: 0.5627 - accuracy: 0.7270 - 246ms/epoch - 919us/step
Loss: 0.5627384185791016, Accuracy: 0.7269970774650574
```

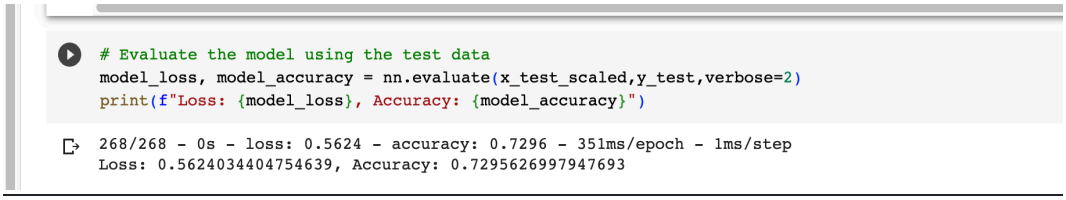
Second try:



```
# 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: 0.5648 - accuracy: 0.7286 - 345ms/epoch - 1ms/step
Loss: 0.5647872090339661, Accuracy: 0.7286297082901001
```

Third try:



```
# 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: 0.5624 - accuracy: 0.7296 - 351ms/epoch - 1ms/step
Loss: 0.5624034404754639, Accuracy: 0.7295626997947693
```

- What steps did you take in your attempts to increase model performance?

  I kept the “name” column in both tries, adjusted the neurons, in the second try I added two additional hidden layers, changed the activation method to tanh, and increased epochs to 200. None of the tries resulted in a higher model performance.

3. **Summary:**

The model did not produce the desired accuracy score given multiple tries. Possibly adding addition hidden layers and additional neurons added to the layers could improve accuracy, however during the second try, that did not produce the desired results either. One recommendation would be the use of a linear regression based model.