Deep Learning Challenge: Venture Capital

Overview:

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With your knowledge of machine learning and neural networks, you'll use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

Results:

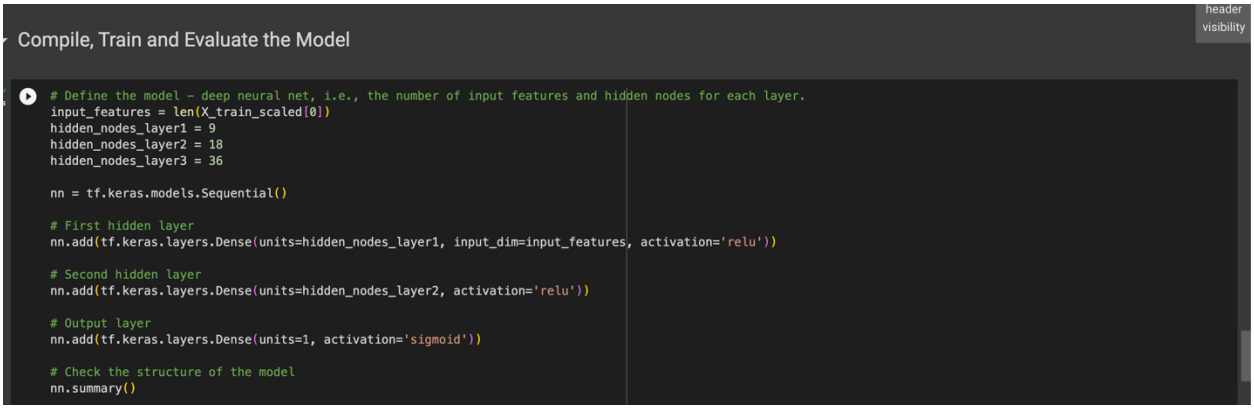
Preprocess steps were taken such as removing not needed columns ['EIN', 'NAME']. The remaining columns were left to be considered for the model.

Split for training and testing sets were performed. The target variable was ['IS_SUCCESSFUL'] with the value of 1 = yes, and 0 = no.

Binned ['APPLICATION_TYPE', 'CLASSIFICATION'] values and used 'cutoffs' to further refine binning. Binning values were encoded by get-dummies.

Compiling, Training, and Evaluating the Model:

There were three layers total for each model after applying Neural Networks.



---

604 parameters were created with 73% accuracy which was below the required 75%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 9)                 405

 dense_1 (Dense)             (None, 18)                180

 dense_2 (Dense)             (None, 1)                 19

=================================================================
Total params: 604 (2.36 KB)
Trainable params: 604 (2.36 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Epoch 490/500
0s
# 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: 0.5639 - accuracy: 0.7291 - 440ms/epoch - 2ms/step
Loss: 0.5638511180877686, Accuracy: 0.7290962338447571
```

Optimization:

The second attempt with changes to binning counts of < 1000, achieved an accuracy of 73%.

```
0s
# 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: 0.5590 - accuracy: 0.7284 - 376ms/epoch - 1ms/step
Loss: 0.5589677691459656, Accuracy: 0.728396475315094
```

The third attempt with changes to binning counts of < 100, achieved an accuracy of 73%.

```
0s
# 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: 0.5541 - accuracy: 0.7292 - 353ms/epoch - 1ms/step
Loss: 0.5540683269500732, Accuracy: 0.7292128205299377
```