Deep Learning Challenge

Overview:

The goal of this project was to create a deep learning model for a non-profit foundation called Alphabet Soup that predicts whether or not the applicants they provide funding for will be successful or not.

Results:

- The target variable for this model is the column “IS_SUCCESSFUL” which consists of ones and zeros. One for successful, zero for unsuccessful.

- There were 9 features used in the model training. These consist of: APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS and ASK_AMT

- The EIN and NAME columns were dropped because they are insignificant factors when it comes to predicting success.

Compiling, Training and Evaluating the Model:

- I was unable to hit the 75% accuracy score, reaching 73% accuracy. After testing different bin sizes, different amounts of layers, many different combinations of nodes, different optimizers, a different scaler and even trying logistic regression, I believe one of the only ways to increase the accuracy is by including the ‘NAME” feature which would cost a lot of time to run.

- I ended up using a model with 881 total parameters and 3 hidden layers.

- I found that changing the amount of layers and nodes had minimal impact on the accuracy score. Using the MinMaxScaler slightly performed better than the StandardScaler.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                704

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 4)                 36

 dense_3 (Dense)             (None, 1)                 5

=================================================================
Total params: 881 (3.44 KB)
Trainable params: 881 (3.44 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
# 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}")
```

✓ 0.6s

```
268/268 - 1s - loss: 0.5515 - accuracy: 0.7312 - 513ms/epoch - 2ms/step
Loss: 0.5515233278274536, Accuracy: 0.731195330619812
```

Summary:

While the model did not reach the desired accuracy, it only missed by ~2%. My only recommendation would be to simply collect more data. I tried using some machine learning models such as Logistic Regression and Random Forest Classifier, but the results were very similar so I can't recommend those.