**Overview**

This project was designed to help create a predictive model to determine whether applicants for aid from the non-profit group Alphabet Soup will successfully use the funding they request.

It uses a neural net deep learning model to make a binary prediction of funding success based on a sample size of approximately 34,000 previous aid grants.

The data was preprocessed to remove irrelevant columns, bin rare variable values, and convert categorical data to numeric for the purpose of modeling.

Tensorflow Keras was used to build and compile a neural net model based on the number of features with the number of layers and neurons set to attempt to generate highly accurate result.

As the initial model fell slightly below 75% accuracy, several further attempts were made to optimize the model and increase its accuracy.

**Variables**

EIN and NAME—Identification columns  
APPLICATION\_TYPE—Alphabet Soup application type  
AFFILIATION—Affiliated sector of industry  
CLASSIFICATION—Government organization classification  
USE\_CASE—Use case for funding  
ORGANIZATION—Organization type  
STATUS—Active status  
INCOME\_AMT—Income classification  
SPECIAL\_CONSIDERATIONS—Special consideration for application  
ASK\_AMT—Funding amount requested  
IS\_SUCCESSFUL—Was the money used effectively

**Results**

* **Preprocessing**

1. The only target variable in the dataset is IS\_SUCCESSFUL.
2. The features which contribute to the analysis include: APPLICATION\_TYPE, AFFILIATION, CLASSIFICATION, USE\_CASE, ORGANIZATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATIONS, and ASK\_AMT.
3. EIN and NAME are both identifications for the specific businesses that received funding in the past. As such, they do not contribute directly to the success of the funding and therefor are neither targets nor features.

* **Compiling, Training, and Evaluating the Model**

1. The model was built with 3 hidden layers because there was a relatively high number of input dimensions, namely 39-43 depending on which setup I was running, with a later modification to 4 layers having little impact on the model's accuracy.  
   For the number of neurons, I went with the rule-of-thumb stating that the number should be less than twice the size of the input layer, with that leading to the number of 80 neurons for the first layer. For the second hidden layer, I went with 30 to be somewhat fewer than the number of inputs. For the output layer I chose 1 to match the number of target dimensions.  
   I used Relu as the method for both 1st hidden layers and sigmoid for the output layer due to being familiar with them from previous experience.
2. The model was unable to reach the target accuracy of 75% in either of my 2 attempts. The peak value was ~74% with all other attempts being virtually identical.
3. For my second attempted a few different methods to increase the performance of my model to reach >75%.
4. First, added 3 hidden layers instead of using 2 like the first attempt.  
   Second, increased the number of neurons in the 1st hidden layer to 100, 2nd hidden layer to 30, & 3rd hidden layer to 10.  
   Third, made the activation for both the second & third hidden layers set to sigmoid.  
   Neither of these methods yielded positive results.

**Summary**

On the whole, while the model never reached the target accuracy of 75%, it did come quite close at 74% for the best iteration.

One issue was the lack of any real impact from the attempted optimization methods for the model. It may have worked better to commit to using Tensorflow's Keras Tuner library to more rigorously examine the effects of varying the hyperparameters.

It is also possible that some of the other feature variables may have been able to be dropped to improve model accuracy. Determining which variables are more valuable to preidicting funding success would take a more rigorous investigation than was possible on the scale of this project.

As for whether some other type of model would be more appropriate for creating a binary classifier for investment success, I can't come up with one.