**OVERVIEW**:

The non-profit foundation Alphabet Soup wants to create an algorithm to predict whether or not applicants for funding will be successful.

Using Machine Learning, Neural Networks and features provided in the dataset I created a binary classifier that is capable of predicting whether applicants will be successful if funded by Alphabet Soup.

The data based on a sample size of approximately 34,000 previous aid grants, was modified to remove both the “EIN” and “NAME” columns. For further Optimization three additional columns were removed, “ORGANIZATION”, “STATUS”, AND “SPECIAL CONSIDERATIONS”.

Additional bin rare variable values were reconfigured for optimization.

Using TensorFlow Keras to build and compile a Neural Networks model based on the number of features with the number of layers and neurons set to attempt to generate highly accurate result and to help further optimize its accuracy.

The initial model fell slightly below 75% accuracy, three optimization attempts were applied to help further/optimize the model and increase its accuracy.

**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 hidden layers and sigmoid for the output layer due to being familiar with them from previous experience. The model was unable to reach the target accuracy of 75% in any of my attempts. The peak value was ~73% with all other attempts being virtually identical.
2. Three different modifications to the were changed to try and increase the performance of my model.  
   Adding another hidden layer between my two original hidden layers with 60 neurons.  
   Doubling the number of neurons in each hidden layer.  
   And dropping three additional columns; ORGANIZATION, STATUS and SPECIAL\_CONSIDERATIONS variables from the data to see if that would help optimize the end analysis.

None of these methods helped optimize an above 75% accuracy.

**SUMMARY:**

On the whole, while the model never reached the target accuracy of 75%, it did come quite close at 73% for the best results for methods used.

Further analysis and methods to using TensorFlow’s Keras Tuner library, Random Forest, Decision Tree to name a few could have been used to help further examine and approve the optimization.