Project 4 - Alphabet Soup Venture Capital

<u>**Overview**</u>

Alphabet Soup has been lauded for its generosity with funding and its knack for finding ventures that can create successful and innovative products and services. However, with so many new ventures and such limited resources for funding and selection, a deep learning model can prove beneficial for Alphabet Soup and its prospective beneficiaries in efficiently and effectively providing capital. Using a dataset of more than 34,000 organizations that have received funding from Alphabet Soup, we can train a model to look at different aspects of a business to determine their eligibility for funding.

<u>**Results**</u>

<u>Preprocessing</u>

After preprocessing, the dataset consisted of the following columns, which were used as features or independent variables:

- APPLICATION_TYPE
- APPLICATION
- CLASSIFICATION
- USE_CASE
- ORGANIZATION
- STATUS
- INCOME_AMT
- SPECIAL_CONSIDERATIONS
- ASK_AMT

… and this list of features would determine the impact of whether or not an organization is successful, helpfully labeled in the dataset as “IS_SUCCESSFUL”.

Because the neural network model can only process numerical data, the “EIN” and “NAME” columns, which are the individual application number and organization name respectively, were dropped from the dataset and the rest of the data were put through a scaler in order to change categorical data into numerical data.

<u>Compiling, Training, and Evaluating the Model</u>

The initial attempt at creating a neural network consisted of two hidden layers, with 30 neurons each and both possessing a rectified linear unit (ReLU) activation function. The hidden layers were given 30 neurons upon the suggestion that the number of hidden neurons should be about ⅔ the size of the input layer, plus the size of the output layer. The ReLU function was chosen as it is handy in modeling input data for classification or regression, without being too computationally taxing. The output layer consisted of a single node with a sigmoid function, in order to gauge the probability of a venture being successful.

Training and testing of this model produced subpar results: the model experienced an end data loss of 0.5566 and a final accuracy of 0.7280. In order to find a more ideal model, TensorFlow's Keras Tuner module was employed in order to test out various neural net arrangements, differing in the number of hidden layers, neurons, and activation functions. After running the Tuner, the best model's hyperparameters were as follows:

```
{'activation': 'tanh',
 'first_units': 36,
 'num_layers': 2,
 'units_0': 16,
 'units_1': 6,
 'units_2': 1,
 'units_3': 21,
 'units_4': 36,
 'units_5': 11,
 'tuner/epochs': 50,
 'tuner/initial_epoch': 17,
 'tuner/bracket': 1,
 'tuner/round': 1,
 'tuner/trial_id': '0075'}
```

This model had more improved performance, although the improvements were minimal: data loss came in at 0.5544 and accuracy was 0.7348.