**Alphabet Soup Funding Predictor**

**Module 21 Challenge**

Zachary Mason

**Background and 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 my knowledge of machine learning and neural networks, I used the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

From Alphabet Soup’s business team, we have received a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. Within this dataset are a number of columns that capture metadata about each organization, such as:

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 considerations for application

ASK\_AMT—Funding amount requested

IS\_SUCCESSFUL—Was the money used effectively

**Results**

The first step of the process involved reading in the data from the csv file and beginning to process it to get it ready for the machine learning model. After processing the data, I compiled it together and developed a neural network, or deep learning model, to create a binary classification model that can predict if a funded organization will be successful based on the features in the dataset.

Data Preprocessing:

* Target Variable:
  + ‘IS\_SUCCESSFUL’
    - Indicates whether the funding was used successfully
* Feature Variables
  + ‘APPLICATION\_TYPE’
  + ‘AFFILIATION’
  + ‘CLASSIFICATION’
  + ‘USE\_CASE’
  + ‘ORGANIZATION’
  + ‘STATUS’
  + ‘INCOME\_AMT’
  + ‘SPECIAL\_CONSIDERATIONS’
  + ‘ASK\_AMT’
* Variable(s) to be removed from the input data because they are neither targets nor features:
  + The ‘EIN’ and ‘NAME’ columns were removed from the input data initially as they serve as identification columns and do not provide predictive value.
* The data was split into training and testing sets and scaled using ‘StandardScaler’
* Variables were encoded using ‘pd.get\_dummies’

Compiling, Training, and Evaluating the Model

* The initial model did not achieve the target accuracy of 75%. In the initial model, the number of neurons, layers, and activation function were utilized to get a base sense of how well the model would perform. In the next section, the selection of those factors were manipulated in or to reach our target goal.

Optimization Steps

1. Increased, decreased the number of neurons on each layer.
2. Added and then later removed additional hidden layers after evaluation.
3. Experimented with different activation functions for the hidden layers.
4. Adjusted the number of epochs to ensure the model was trained adequately.

**Summary**

After the first few attempts at optimizing the model failed to reach the target goal of 75%, only adjusting the neurons, activator, and layers, I decided to go another route. This time around, in the preprocessing stage I decided to keep the ‘NAME’ variable as a feature and altered/binned the name of any company with less than 5 instances as “Other.” Once that was added back in, we were able to achieve an accuracy of 79% and loss of 45% where the accuracy was roughly 4% higher than the goal.

One option to further improve the model would be to use Keras Tuner to automate the optimization of the deep learning model. This will help one select the best model configuration by searching through for the most ideal combination of parameters (i.e. layers, neurons, activation functions, etc..).

**AlphabetSoupCharity\_Optimization\_4**

A black screen with white text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A computer screen with text on it

Description automatically generated

A screenshot of a computer program

Description automatically generated