Neural Network Model Report

# Overview

I 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. I was tasked with building 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.

# Data Processing

To process the data, I removed unnecessary columns (‘EIN’). I then binned ‘NAME’, ‘APPLICATION\_TYPE’, and ‘CLASSIFICATION’. Once that was done, I use pd.get\_dummies on the rest of the columns and set ‘IS\_SUCCESSFUL’ to be the target for the model. The remaining columns were set to be features. The data was then split into training and testing and StandardScaler was used.

# Compiling, Training, and Evaluating the Model

For my second attempt at evaluating the model, I used two hidden layers and one output layer. The first was a relu activation with 10 units. The second was another relu activation with 16 units. And the output layer was a ‘sigmoid’ activation with 1 unit.

I was able to achieve the target model performance, achieving a 79% accuracy. This is 7% higher than my first attempt, which achieve a 72% accuracy.

To increase the model’s performance, I included the ‘NAME’ column, which was previously dropped from the database, and binned it along with ‘APPLICATION\_TYPE’, and ‘CLASSIFICATION’. I played around with the number of hidden layers and units to use. Ultimately, I settled on using three more units on my first hidden layer, and two more units on my second hidden layer than I did previously on my first attempt.

## First Attempt:

Table

Description automatically generated



## Second Attempt:

Table

Description automatically generated



# Summary

In conclusion, the second model performed better all-around at a 79% accuracy and 0.45 loss. Whereas the first model had a 72% accuracy with a 0.55 loss. My recommendation would be to use the second model. In the future I would use an automated neural network to determine the best model hyperparameters to use.