Alphabet Soup Model Report

Overview

The purpose of the created model is to be utilized as a tool to predict if funding applicants for the nonprofit Alphabet Soup will be successful. A CSV of the funding results of 34,000 applicants was provided to train the model. The first or original model predicted success with 73.01% accuracy and the goal with the optimized model was to achieve an accuracy above 75%. Using my knowledge of machine learning and neural networks, I enacted three changes to the first model to enhance performance. The report will discuss why the modifications improved the neural network model and how the use of a different model could also solve the problem.

Results

First Model:

- The first model dropped columns that were deemed "non-beneficial" as they did not need to be included in the input data including EIN and name of organization (NAME).
- After investigating the number of unique values in each column, the application type and government organization classification needed bins for the rare occurrences.
- Categorical data was converted to dummy/indicator variables.
- The target array was the column determing applicant success, "IS_SUCCESSFUL", while the remaining dataset were kept as features.
- Two hidden layers were used with ReLU activation with 80 and 30 neurons. The output layer with 1 neuron used sigmoid activation. There were 5981 total params.
- The model was trained with 100 epochs.
- Accuracy: 0.7301
- Loss: 0.5605

Modifications to the first model to increase performance:

1. The column "NAME" was retained as a feature.
2. Due to the vast number of organization names, the rare occurrences were binned together. Rare occurrences were deemed any organization with less than 50 mentions.
3. The number of epochs was decreased from 100 to 50.

Results of optimized model:

- Model accuracy improved from 73.01% to 76.47%
- Model loss was improved/minimized from 0.5605 to 0.4856.

Summary

Overall, enacting the changes for the optimized model led to a better model performance. However, using a different machine learning model could also solve the problem or even further improve on the optimized model. For instance, random forest could be used in place of a neural network. The model can also be used for classification and offers a similar level of accuracy. Random forest can handle both categorial and numerical data as well as providing high accuracy even with a large set of features. The algorithm would be able to tackle this dataset and classification problem with a similar efficiency to neural networks and testing should be done to see if random forest improves performance.