Alphabet Soup Funding Final Analysis

Purpose

The purpose of the project is to create a model to predict the success of future applicants receiving Alphabet Soup funding. To do so I will analyze a dataset of over 34,000 organizations previously funded by Alphabet Soup, using machine learning and neural networks, to continually improve the model's accuracy.

Analysis

After testing four models, while using different optimization techniques, I was unable to achieve a model accuracy of 75%. Each model achieved between 72% to 73% accuracy with varying degrees of loss.

The best result I received for accuracy was the 1st model with a 72.75% accuracy. The model also resulted in a 55.85% loss. The worst model for accuracy was the 4th model with 72.2% accuracy, and a 57.8% loss.

The best and worst result I received for loss was also the 1st model and 4th model with accuracy and loss percentages as indicated above.

Overall, the best performing model was the 1st model which had the best balance of accuracy and loss. Subsequent models did not improve on the original results.

Data Preprocessing

- What variable(s) are the target(s) for your model?
  - The target for the model was the IS_SUCCESSFUL column.
- What variable(s) are the features for your model?
  - The available features for my model are: EIN, NAME, APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, and ASK_AMT
- What variable(s) should be removed from the input data because they are neither targets nor features?
  - In the first model, as instructed, I removed the EIN column and the NAME columns. In subsequent models, I removed the ORGANIZATION and the INCOME_AMT columns as I did not believe they impacted the success of the campaign.

Compiling, Training, and Evaluating the Model

- How many neurons, layers, and activation functions did you select for your neural network model, and why?
    - In the first model, I used 2 hidden layers with 80 neurons in the first and 30 neurons in the second layer, each utilizing the ReLU activation. For the output layer the Sigmoid activation was used. I believed that ReLU and Sigmoid were the best choices as we were working with a non-linear data set and looking for a binary result. I attempted to use LeakyReLU in subsequent models because I noticed that the accuracy and loss did not improve much between the first epoch to the last, however, I received an error that I was unable to debug and thus abandoned this approach.

    - In the second model, I removed the Organization and Income Amt columns because I thought they were not essential. In hindsight, I should have measured the importance of each column in the calculation before making such adjustments. Additionally, I should have returned to the original model before applying other optimization methods.

    - In my third model, I reduced the number of neurons in my first and second hidden layers. I also reduced the number of epochs. For a better result, I could have added an additional hidden layer and fiddled with the number of neurons in each to see what achieved the best accuracy and loss balance.

    - In my last model, I tried using different activations for the hidden and output layers and settled on the Tanh activation on the output layer, which is also a non-linear activation. This did not achieve the results I hoped for and acyually produced a model with the worst accuracy and loss.
- Were you able to achieve the target model performance?
    - No.
- What steps did you take in your attempts to increase model performance?
    - I tried other activation methods, like Leaky Relu and Tanh, removed columns, changed number of neurons in the layers and number of epochs