**Beautiful Soup Neural Network Model Analysis**

**Overview:**

Alphabet Soup, a fictional, non-profit foundation, has asked us to provide a tool to streamline their funding process by identifying the applicants most likely to succeed in their ventures. Alphabet Soup’s business team has provided us with a comprehensive dataset encompassing over 34,000 organizations. This dataset serves as a rich tapestry of information, detailing varied attributes of each organization such as their Name, Affiliation, Classification, Use Case, Active Status, Income Classification, the Funding Amount they requested, and crucially, the success rate of these requests.

Harnessing the power of machine learning and neural networks, our aim is threefold: analyze and process the dataset, develop and train the model for the neural network, and validate and test the model. At the culmination of this project, we aim to provide Alphabet Soup with a reliable tool that can predict whether an applicant will successfully utilize the funds. By the end of this project, our goal is to give Alphabet Soup a practical tool to help predict if an applicant will effectively use the granted funds. This tool will assist Alphabet Soup in making informed, data-backed decisions regarding their funding allocations.

**Steps and Results:**

*Data Preprocessing and Analysis:*

Machine learning models thrive when fed with high-quality data. The preprocessing of this data is a foundational step that sets the tone for the entire modeling process. Without a meticulous approach to preprocessing, the accuracy and reliability of even the most advanced models can be compromised. This emphasizes the need for our data to be clean, relevant, and properly structured right from the onset.

In our dataset for this project, the target variable is “IS\_SUCCESSFUL”, a binary representation where a value of 1 indicates successful utilization of funds, and 0 indicates otherwise. Two columns, “EIN” and “NAME” are being omitted from our analysis. Their purpose is primarily identification, and they don't contribute to our model's predictive capabilities.

In preparing the data for our neural network model, we found that the dataset contained a rich array of features that captured essential information about the applicants and their requests. The variables captured a variety of details such as the application type, industry sector affiliation, government organization classification, intended use of funds, the organization's active status, type and income range, special considerations for application, and the amount of funding requested. These variables, each with unique characteristics and distributions, necessitated distinct preprocessing approaches. Through iterative model refinement, the following manipulations emerged as the most effective in optimizing our neural network's performance. By applying a range of manipulation strategies, we aimed to enhance the quality and relevance of the input data, and to better equip our neural network to draw meaningful and reliable inferences. The following is a detailed list of the variables and the specific preprocessing steps we employed to ensure their optimal integration into the neural network:

* Application Type: this data was heavily skewed to “T3” as shown in the graph below, however many other application types do have measurable data. Using a threshold of 700 applications, any application type with less than 700 was moved to the ‘Other’ category.

A graph with different colored bars

Description automatically generated

* Classification Type: similar to the distribution observed for the application types, the distribution of classification types exhibited a strong skew towards C1000. The chart below represents classification value counts greater than 1. Initially, we grouped classifications with counts less than 100 into an 'Other' category. However, this approach resulted in a distribution where the 'Other' category had only 669 counts, and it did not sufficiently reduce the number of classification types for effective modeling. After experimenting with different cutoff values, we found that grouping classifications with fewer than 750 counts into the 'Other' category was optimal, as it better balanced the distribution and improved model performance.

A graph of a number of numbers

Description automatically generated

* Use Case: 'Preservation' and 'ProductDev' were the most common values, while 'CommunityServ', 'Heathcare' (this typo is in the dataset), and 'Other' had relatively low counts. To simplify the feature for modeling, we grouped the less common values ('CommunityServ', 'Heathcare', and 'Other') into a single 'Other' category. This grouping reduced the number of unique values in the 'USE\_CASE' column and made the data more suitable for analysis in our neural network model.
* Organization: many entries were either 'Trust' or 'Association'. There were only a few entries for other types such as 'Co-operative' and 'Corporation'. Given the large disparity in value counts, we decided to only consider entries labeled as 'Trust' or 'Association' and removed all other entries from the dataset. This decision was made to focus on the more representative categories and to create a more balanced dataset for our analysis.
* Status: almost all values in this variable are ‘1’, while only a small portion are ‘0’. A variable that is predominantly skewed towards a single value can create an imbalance in the data and may not contribute meaningfully to the model's predictive power. Moreover, it can potentially introduce bias and reduce the generalization ability of the model to unseen data. This value was dropped from our final model.
* Income Amount: upon examining the value distribution, we noticed that most of the applicants had an income amount of '0', followed by other income ranges. To prepare this categorical variable for the neural network model, we used one-hot encoding, a technique that transforms categorical data into a binary matrix, a format more easily understood by our machine learning model[[1]](#footnote-1).
* Special Considerations: this variable is heavily imbalanced, with most of the applicants (33,634) having no special considerations ('N') and only a small portion (27) having special considerations ('Y'). In an earlier model iteration, we considered dropping this variable due to its imbalance. However, upon further analysis, we decided to retain it in the final model, as it seemed to have some relevance to the dataset.
* Ask Amount: Upon examining the distribution of values in this variable, we noticed a strong skew towards a specific amount of $5,000, with 25,142 applicants requesting this amount. The remaining values exhibited a high degree of variability and lower frequencies. We considered categorizing the 'ASK\_AMT' variable into bins based on the amount requested (e.g., '0-10,000', '10,001-50,000', etc.) to manage this high variability. However, this approach did not significantly improve model accuracy, prompting us to explore alternative methods. Given the heavy skew towards the $5,000 value, we decided to create a new binary variable called 'IS\_5000', which indicates whether the requested amount is $5,000 (1) or not (0). This new binary variable was created to specifically address the disproportionate representation of the $5,000 request amount in the dataset. By capturing this unique characteristic, we aimed to provide the model with an additional feature that could potentially help distinguish patterns or trends associated with the prevalence of this particular funding request amount. Next, we applied a logarithmic transformation to the 'ASK\_AMT' variable using the 'np.log1p' function (logarithm of 1 plus the input value) to manage the wide range of values and reduce skewness. Finally, we scaled the transformed 'ASK\_AMT' variable using Min-Max scaling. This scaling technique transforms the values to fall within a specified range (usually 0-1), ensuring that the 'ASK\_AMT' variable does not disproportionately influence the model due to its large numerical values.

*Compiling, Training, Evaluation of Model:*

Before a neural network begins its analysis, it must be properly set up. Compiling the model involves establishing the mathematical and computational framework that dictates how the model will update and adjust itself during training. This step encompasses decisions about which optimizer to use, how loss is calculated, and which metrics to monitor.

In the neural network model, we used two hidden layers and one output layer. The architecture is as follows:

* First Hidden Layer: 512 neurons, ReLU activation function
* Second Hidden Layer: 448 neurons, ReLU activation function
* Output Layer: 1 neuron, Sigmoid activation function

We chose the ReLU activation function for the hidden layers because it is widely used in deep neural networks and is effective at introducing non-linearity into the model. It is computationally efficient and helps the model learn more complex relationships in the data. The sigmoid activation function was selected for the output layer to squash the output values into the range [0, 1], making it suitable for binary classification tasks.

The choice of 512 neurons for the first hidden layer and 448 neurons for the second hidden layer was based on experimentation and fine-tuning. The goal was to strike a balance between model complexity and performance, and these configurations proved to be effective.

In the initial iterations, we tried using 416 and 480 neurons in the first hidden layer and 480 and 96 neurons in the second hidden layer. However, based on the Hyperband optimization results, we adjusted the number of neurons to 512 and 448 in the first and second hidden layers, respectively. These changes were informed by the Hyperband search, which found these configurations to yield better model performance in different runs.

We also applied several preprocessing steps to the input features, including encoding categorical variables, binning and transforming the 'ASK\_AMT' feature, and scaling the features. These preprocessing steps were designed to transform the raw data into a format suitable for training a neural network and help improve the model's predictive performance. Notably, the Hyperband results revealed that the best model performance was achieved with minimal preprocessing in the first set, guiding further preprocessing choices.

Hyperband optimization revealed that the learning rates of 0.0001 or 0.001 were selected as the best values across multiple runs, suggesting that smaller learning rates lead to more stable and gradual convergence. We took this into account when compiling and training the neural network. To assess the model's capability, it was evaluated on previously unseen data, ensuring it was not merely memorizing the training data but effectively generalizing to novel situations. This iterative process of compiling, training, and evaluating solidified our confidence in the model's practical applicability, confirming it is both mathematically robust and useful in real-world scenarios.

**Summary:**

In summary, the deep learning model we implemented used two hidden layers and one output layer, with 512 and 448 neurons in the first and second hidden layers, respectively. We chose the ReLU activation function for the hidden layers and the sigmoid activation function for the output layer. We applied several preprocessing steps to the input features, including encoding categorical variables, binning and transforming the 'ASK\_AMT' feature, and scaling the features. The model architecture and preprocessing steps were informed by Hyperband optimization results, which suggested that smaller learning rates of 0.0001 or 0.001 were more effective for stable and gradual convergence.

The overall performance of the model varied depending on the preprocessing applied to the input features and the chosen architecture. In our experiments, the best performance achieved was 73.38% on the base set with no significant changes from the initial dataset other than binning. It is essential to note that the Hyperband results showed the best model performance occurred with minimal preprocessing in the first set, which provides a valuable insight into potential preprocessing choices in the future.

Despite achieving promising results with the deep learning model, there are other potential approaches to solving this classification problem. One alternative model could be a Random Forest classifier, a powerful ensemble learning technique. Random Forests consist of multiple decision trees, which are combined to produce more accurate and robust predictions. Each tree is trained on a random subset of the data and features, reducing the likelihood of overfitting and increasing model diversity.

In conclusion, the deep learning model showed promise in classifying the success of Alphabet Soup's business proposals. However, an alternative approach, such as a Random Forest classifier, could offer a more robust and interpretable solution to this classification problem. Further experimentation and comparison of these models would help determine the most effective approach for Alphabet Soup's specific needs.

1. Multiple other categorical variables in the dataset, including 'APPLICATION\_TYPE', 'AFFILIATION', 'CLASSIFICATION', 'USE\_CASE', 'ORGANIZATION', ‘ASK\_AMT’, and 'SPECIAL\_CONSIDERATIONS' were also one-hot encoded. [↑](#footnote-ref-1)