**Funding Success Prediction: A Deep Learning Approach**

**Overview**

The purpose of this task is to develop a binary classifier using machine learning and neural networks to assist the nonprofit foundation Alphabet Soup in selecting applicants for funding with the best chance of success in their ventures.

With the provided dataset containing information on over 34,000 organizations that have received funding from Alphabet Soup, the goal is to create a predictive model that can accurately classify whether an applicant will be successful if funded.

By leveraging machine learning techniques and neural networks, analyze these features to build a model that can effectively predict the success of funding applicants. This model will enable Alphabet Soup to prioritize funding for organizations that are more likely to utilize the funds effectively, maximizing the impact of their investments and resources.

**Results**:

Data Preprocessing

* The Target variable is “IS\_SUCCESSFUL” for this model
* The features variables for this model are APPLICATION\_TYPE , AFFILIATION, CLASSIFICATION, USE\_CASE, ORGANIZATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATIONS, and ASK\_AMT

EIN and Name variable should be removed from the input data because they are neither targets nor features, there identification columns.

**Compiling, Training, and Evaluating the Model:**

For the neural network model, I have selected the following configuration:

Number of neurons:

* First hidden layer: 80 neurons
* Second hidden layer: 30 neurons

Number of layers:

* Three layers in total (two hidden layers and one output layer)

Activation functions:

* ReLU (Rectified Linear Unit) for hidden layers
* Sigmoid for the output layer

**Reason for selecting these parameters:**

Number of neurons:

The number of neurons in each layer is typically chosen based on the complexity of the problem and the size of the dataset. Since we have 43 input features, I chose 80 neurons for the first hidden layer to allow the network to learn complex patterns from the data. The second hidden layer has 30 neurons to progressively reduce the number of features and extract higher-level representations.

Number of layers:

I selected a total of three layers to capture the complexity of the data. Adding more layers can increase the model's capacity to learn intricate patterns, but too many layers may lead to overfitting, especially with limited data.

Activation functions:

ReLU (Rectified Linear Unit) is commonly used in hidden layers because it allows the network to learn complex nonlinear relationships in the data. It is computationally efficient and helps alleviate the vanishing gradient problem.

Sigmoid activation function is used in the output layer for binary classification tasks because it squashes the output values between 0 and 1, representing the probability of the positive class.

**Were you able to achieve the target model performance?**

The result with the target model is

Accuracy: 72.62%

Loss:0.582

The target performance was not achieved. The goal was to achieve a higher accuracy, ideally around 75% or higher.

**Steps taken to increase model performance**

The AlphabetSoupCharity model achieved a slightly higher accuracy compared to the first model ( from 72.62 to 73.24%). But still not managed to achieved a 75% accuracy. Some of the changes made to the AlphabetSoupCharity model to achieve the improvement are:

Feature Engineering:

The AlphabetSoupCharity model performed feature binning on the 'CLASSIFICATION' column by replacing less frequent classifications with 'Other'( cutoff=1000), reducing the number of unique categories. This feature engineering step likely helped in reducing noise and focusing the model on the most relevant categories.

Number of Input Features:

The AlphabetSoupCharity model had fewer input features (29) compared to the first model (43) due to the feature binning process. This reduction in input features could have simplified the model and made it easier for the neural network to learn the patterns in the data.

**Model Architecture:**

The AlphabetSoupCharity model had a deeper architecture with more hidden layers:

First hidden layer: 80 neurons

Second hidden layer: 50 neurons

Third hidden layer: 40 neurons

Fourth hidden layer: 40 neurons

Fifth hidden layer: 30 neurons

This deeper architecture allowed the model to capture more complex patterns in the data.

**Optimization Algorithm and Learning Rate:**

The AlphabetSoupCharity model used a custom learning rate of 0.0001 with the Adam optimizer. Adjusting the learning rate can impact the convergence and performance of the model. Lowering the learning rate could have helped the model converge more effectively.

**Summary**:

while the deep learning model achieved a reasonable accuracy, trying a Random Forest Classifier could provide a simpler, more interpretable solution with potentially comparable performance. Random Forest models are generally easier to train and tune compared to deep learning models. They are less sensitive to hyperparameters and require less computational resources. This approach would be beneficial for Alphabet Soup's business team to understand the factors influencing funding decisions and could provide insights into which organizations are most likely to be successful with funding.