Neural Network Model for Alphabet Soup

Overview of the Analysis

The main objective of this study is to develop a deep learning-based neural network model that can better predict whether alphabet soup funding organizations will effectively use the funds allocated to their projects. Alphabet Soup, a non-profit organization, seeks to optimize its fund allocation by identifying applicants most likely to have the greatest success. To achieve this goal, a binary classification model was used that predicts the success or failure of funded organizations based on a dataset with detailed information about past funding recipients.

Results

Data Preprocessing

1. Target Variable(s): The primary target variable for the model is "IS_SUCCESSFUL," which is a binary indicator of whether the funding was effectively utilized (1 for success, 0 for failure).

2. Features for the Model: A set of features was selected to serve as input for the model. These features encompass various aspects of the applicant organizations and their funding requests. The selected features include "APPLICATION_TYPE," "AFFILIATION," "CLASSIFICATION," "USE_CASE," "ORGANIZATION," "INCOME_AMT," and "ASK_AMT."

3. Variables Removed: During the data preprocessing phase, a number of variables were identified and removed as they were considered irrelevant to the model building process: "EIN" and "NAME". These columns were removed because they were not expected to provide significant insights into the prediction.

4. Standardized the data using StandardScaler.

Compiling, Training, and Evaluating the Model

Several iterations were made to test different configurations for the neural network model. These iterations involved modifying the number of neurons and layers within the network.

1. Original Model (Before Optimization)
   Purpose: The purpose of the first model was to develop a baseline for predicting the effectiveness of funding based on the provided dataset.

   Neural Network Architecture:
   Input Layer:

First Hidden Layer: 10 neurons with ReLU activation.
Second Hidden Layer: 5 neurons with ReLU activation.
Output Layer: 1 neuron with sigmoid activation (binary classification).

2. First Optimization Attempt
   Purpose: The first optimization aimed to enhance model performance by adjusting the neural network architecture.

   Neural Network Architecture:
   First Hidden Layer: Increased to 40 neurons with ReLU activation.
   Second Hidden Layer: Added a second hidden layer with 20 neurons, both with ReLU activation.
   Output Layer: 1 neuron with sigmoid activation (binary classification).

   Data Preprocessing:
   Modified cutoff values for "APPLICATION_TYPE" (500) and "CLASSIFICATION" (1000) to manage infrequent categories.
   Standardized the data using StandardScaler.

3. Second Optimization Attempt
   Purpose: The second optimization aimed to further improve model performance by tweaking the architecture and preprocessing.

   Neural Network Architecture
   First Hidden Layer: Maintained 40 neurons with ReLU activation.
   Second Hidden Layer: Kept the second hidden layer with 20 neurons and ReLU activation.
   Output Layer: 1 neuron with sigmoid activation (binary classification).

   Data Preprocessing:
   Adjusted cutoff values for "APPLICATION_TYPE" (800) and "CLASSIFICATION" (1500) to address infrequent categories.
   Standardized the data using StandardScaler.

4. Third Optimization Attempt
   Purpose: In the third optimization the objective was to explore the impact of removing additional non-beneficial columns and adjusting the neural network architecture. "STATUS" and "SPECIAL_CONSIDERATIONS" were removed because they were not expected to provide significant insights into the prediction of financial efficiency

   Neural Network Architecture:
   First Hidden Layer: Retained 40 neurons with ReLU activation.
   Second Hidden Layer: Preserved the second hidden layer with 20 neurons and ReLU activation.

Third Hidden Layer: Added a third hidden layer with 5 neurons and sigmoid activation.
Output Layer: 1 neuron with sigmoid activation (binary classification).

Data Preprocessing:
Dropped more non-beneficial columns ("EIN," "NAME," "STATUS," "SPECIAL_CONSIDERATIONS") to streamline input data.
Adjusted cutoff values for "APPLICATION_TYPE" (500) and "CLASSIFICATION" (1000) to manage infrequent categories.
Standardized the data using StandardScaler.

Steps Taken for Model Performance Improvement

Throughout the optimization attempts, adjustments to the neural network's architecture were made, including the number of neurons and layers.

Various activation functions were used, including ReLU and sigmoid.

The cutoff values for categorical variables like "APPLICATION_TYPE" and "CLASSIFICATION" to manage infrequent categories were also changed.

Unfortunately, despite the efforts, the target model performance of 75% accuracy was not met. The most successful model achieved an accuracy of approximately 72.50% when evaluated on the testing dataset.

Summary

Our deep learning neural network model exhibits potential for predicting the effectiveness of funding for organizations; however, further optimization is required to achieve the target 75% accuracy. Several attempts were made to try to optimize the model ncluding varying the number of neurons and the number of activating functions, but the level of accuracy was not achieved.

Recommendation for a Different Model

Random Forest could be used as a recommendation for this example as it is a robust and versatile ensemble learning model that can be well-suited to solve this classification problem. Main strengths of this model are:

Ensemble Power: learning technique that combines multiple decision trees to make predictions. This ensemble approach often results in more accurate and stable predictions.

Categorical Data Handling: such as application types, without requiring extensive preprocessing. It simplifies the data preparation process.

Feature Importance: provides insights into feature importance. It helps identify which variables play a significant role in making predictions, akin to understanding which factors drive outcomes.

Model Simplicity: relatively easier to interpret compared to complex models like deep neural networks. It employs a collection of simple decision trees, making it more comprehensible.

Robustness to Noise: robust and less sensitive to outliers and noisy data, making it suitable for real-world datasets with imperfections.

Hyperparameter Tuning: offers numerous hyperparameters that can be fine-tuned to optimize performance. Techniques like grid search or random search can help find the best combination of hyperparameters.