**Alphabet Soup Deep Learning Model Report**

**Overview of the Analysis**

The goal of this analysis was to create a deep learning model that predicts if an applicant to Alphabet Soup, a non-profit organization, will become a successful donor. By using data from past applications, we built a binary classification model to help the organization focus on the most promising applicants.

**Results:**

**Data Preprocessing**

* **Target Variable:**
  + IS\_SUCCESSFUL: This is the column we want to predict. It shows if the applicant became a successful donor (1 = Yes, 0 = No).
* **Feature Variables:**
  + All the other columns after converting the data to numbers. These include:

APPLICATION\_TYPE, AFFILIATION, CLASSIFICATION, USE\_CASE, ORGANIZATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATIONS, and ASK\_AMT.

* **Removed Variables:**
  + EIN and NAME were removed because they are ID numbers or names and don’t help the prediction.
* **Steps Taken:**
  + Categories in APPLICATION\_TYPE and CLASSIFICATION that had very few records were replaced with the label “Other.”
  + We used pd.get\_dummies() to change the categorical variables into numbers.
  + We split the data into training and testing sets.
  + Then, we used StandardScaler to scale the feature data.

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

* **Neural Network Architecture:**
  + **Input Features:** 43 columns
  + **Hidden Layer 1:** 80 neurons, ReLU activation
  + **Hidden Layer 2:** 30 neurons, ReLU activation
  + **Output Layer:** 1 neuron, Sigmoid activation (because it’s a binary classification)
* **Training Settings:**
  + Optimizer: Adam
  + Loss Function: Binary Crossentropy
  + Epochs: 100
* **Model Performance:**
  + **Final Accuracy:** **72.59%**
  + **Final Loss:** 0.5557
* **Steps to Improve Accuracy:**
  + Grouped rare values under “Other” to reduce noise.
  + Normalized the numeric data using a scaler.
  + Chose ReLU activation to avoid the vanishing gradient problem.
  + Trained the model for 100 epochs.

**Summary and Recommendation**

The final model reached an accuracy of **72.59%**, which is close to the target of 75%. The model showed that deep learning can give helpful predictions for this kind of data. However, there is still room to improve the results.

**Recommendation:**

We suggest trying a different model like **XGBoost** or **Random Forest**. These models usually work better with structured data (tables with categories and numbers), and they often give higher accuracy without much tuning.

**Why:**

Neural networks are powerful but can be harder to train with this kind of data. Tree-based models, like XGBoost, are faster to train and can find important patterns easily. They can also show which variables are most important in predicting success.