# Report on the Neural Network Model for Alphabet Soup

---

**Overview of the Analysis**

**The purpose of this analysis is to develop a deep learning binary classification model to predict whether an organization funded by Alphabet Soup will be successful or not, based on historical data. The goal is to accurately predict the outcome (`IS_SUCCESSFUL`) using various features related to the organization's characteristics and funding.**

---

**Results**

**Data Preprocessing**

**- Target Variable:**

**- The target variable (`y`) for the model is `IS_SUCCESSFUL`, which indicates whether an organization was successful (1) or not (0).**

**- Feature Variables:**

**- The features used for the model are all other variables that provide insight into the organizations, such as:**

**- `APPLICATION_TYPE`**

**- `AFFILIATION`**

**- `CLASSIFICATION`**

**- `USE_CASE`**

**- `ORGANIZATION`**

**- `STATUS`**

**- `INCOME_AMT`**

**- `ASK_AMT`**

**- `SPECIAL_CONSIDERATIONS`**

**- Variables Removed:**

**- The following columns were removed as they are not useful for predicting the success of an organization:**

**- `EIN` (Employer Identification Number)**

**- `NAME`**

**These variables were removed because they are identifiers, and including them in the model would not contribute to its predictive power.**

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

**- Neurons, Layers, and Activation Functions:**

**- Input Layer: The input layer used the number of features after preprocessing, which was the number of columns in the dataset after dropping the unnecessary ones.**

**- First Hidden Layer: The model initially used 128 neurons with the ReLU activation function. This layer is responsible for detecting non-linear patterns in the data.**

**- Second Hidden Layer: The second layer had 64 neurons, also using the ReLU activation function to continue extracting non-linear relationships from the data.**

**- Third Hidden Layer: The second layer had 32 neurons, also using the ReLU activation function to continue extracting non-linear relationships from the data.**

**- Output Layer: The output layer used 1 neuron with the sigmoid activation function, which is ideal for binary classification problems.**

Compile, Train and Evaluate the Model

```
[ ] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
    number_input_features = len(X_train_scaled[0])
    hidden_nodes_layer1 = 128
    hidden_nodes_layer2 = 64
    hidden_nodes_layer3 = 32

    nn = tf.keras.models.Sequential()

    # First hidden layer
    nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

    # Second hidden layer
    nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

    # Third hidden layer (new layer)
    nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation='relu'))

    # Output layer
    nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

    # Check the structure of the model
    nn.summary()
```

**Reasoning:**

**- The ReLU (Rectified Linear Unit) activation function was selected because it helps prevent the vanishing gradient problem, making it a standard choice for hidden layers in deep learning models.**

**- Sigmoid was chosen for the output layer to ensure the output is a probability between 0 and 1, suitable for binary classification.**

**- Model Performance:**

**- The model was trained for 100 epochs, and the final accuracy on the test data was 72.95%, with a loss of 0.636. While this performance is reasonable, it did not meet higher performance targets, indicating room for improvement.**

**- Steps Taken to Improve Performance:**

**- Several steps were taken to increase model performance:**

**- Added Dropout layers: Dropout regularization (0.2) was introduced to reduce overfitting by randomly deactivating neurons during training.**

**- Batch Normalization: This was added to normalize the activations of each layer, stabilizing the training process and allowing for faster convergence.**

**- Early Stopping: Early stopping was implemented to halt training when validation loss stopped improving, which helps to prevent overfitting from too many epochs.**

**- Optimizer Tuning: The Adam optimizer was used with a tuned learning rate, which generally works well for deep learning models.**

---

**Summary**

**- Overall Results:**

**- The model achieved a 72.95% accuracy on the test data with a loss of 0.636 after 100 epochs. The performance is reasonable but shows signs of overfitting, as the training accuracy was higher than the test accuracy.**

**- While the model detected patterns in the data, improvements in regularization, model architecture, or alternative approaches could lead to better performance.**

**- Recommendation for Alternative Models:**

**- Random Forest or Gradient Boosting models could provide better accuracy and stability for this binary classification problem. These models often perform well with structured/tabular data like the one provided, and they are less prone to overfitting compared to neural networks in smaller datasets. Moreover, these models can provide feature importance metrics, offering insights into which features have the greatest impact on predicting success.**

**- Why Tree-Based Models?**

**- They are robust with small to medium-sized datasets.**

**- They handle imbalanced datasets and categorical data well.**

**- They can capture complex relationships without requiring as much tuning as neural networks.**