**Predicting Funding Success for Alphabet Soup Nonprofit Foundation Report**

**Analysis Overview**

The objective of this analysis is to build a binary classifier that predicts whether an applicant funded by Alphabet Soup will be successful in their ventures. The analysis is based on a dataset of over 34,000 organizations that have previously received funding. By using machine learning techniques, particularly a neural network, this model will help the nonprofit select applicants who have the best chance of success, thus optimizing resource allocation. This assignment involves three parts:

**PART 1: Data Cleaning**

The dataset contains several columns with metadata about organizations, including: The “**EIN”** and “**NAME”** columns serve as identification fields. Categorical variables such as “**APPLICATION\_TYPE”**, “**AFFILIATION”**, “**CLASSIFICATION”**, “**USE\_CASE”**, “**ORGANIZATION”**, and “**STATUS”** provide information about the nature and status of each organization. Financial details are captured in the “**INCOME\_AMT”**, “**SPECIAL\_CONSIDERATIONS”**, and “**ASK\_AMT”** columns. The target variable, “**IS\_SUCCESSFUL”**, indicates whether the funding provided to the organization led to a successful outcome.

* **Target Variable:** The target variable for the model is “**IS\_SUCCESSFUL”**, which shows whether a funded project was successful.
* **Columns Selection:** The columns “**EIN”** and “**NAME”** are dropped because they do not contribute meaningful information to the model.
* **Handling Unique Values:** For categorical columns, rare categories are grouped under "Other" based on a chosen cutoff point. This reduces dimensionality and helps the model generalize better.
* **Encoding Categorical Variables:** Categorical variables are transformed into numerical values using one-hot encoding through `**pd.get\_dummies()`**.
* **Data Splitting:** The data is divided into training and testing sets using `**train\_test\_split()`**, ensuring a portion of the data is reserved for model evaluation.
* **Data Scaling:** The features in both the training and testing sets are normalized using `**StandardScaler()`**, which ensures the neural network can learn effectively without being affected by varying feature magnitudes.

**PART 2: Neural Network Compilation, Training, and Evaluation**

* **Neural Network Model:** The model is designed using **TensorFlow** and **Keras** to handle the binary classification task. The model consists of:

1. **Input Layer:** The number of input features is equal to the number of columns after encoding and dropping unnecessary fields.
2. **Hidden Layers:**

* The first hidden layer contains a significant number of neurons (80 units), with a `**ReLU`** activation function to introduce non-linearity and capture complex patterns.
* A second hidden layer is added with a reduced number of neurons to compress the features and further refine the network's learning.

1. **Output Layer:** The output layer uses a single neuron with a sigmoid activation function, suitable for binary classification as it provides output probabilities between 0 and 1.

* **Model Training:** The model is compiled using the “**binary\_crossentropy”** loss function, appropriate for binary classification problems, and the “**Adam”** optimizer, which efficiently handles gradient updates. The model is trained on the training set for a specific number of `**epochs`**.
* **Model Evaluation:** The model is evaluated using test data to assess its performance based on loss and accuracy metrics.

**PART 3: Model Optimization**

To achieve a predictive accuracy higher than 75%, several optimization strategies are applied:

* **Increasing Neurons**: More neurons are added to the hidden layers to increase the model's capacity.
* **Adding Hidden Layers**: An extra hidden layer is introduced to allow the model to learn deeper features.
* **Changing Activation Functions**: activation functions are used in different layers to see if they improved performance.
* **Adjusting Training Epochs**: The number of training epochs is adjusted.

After applying various optimizations, the model is re-evaluated using the test set to assess its performance. The primary objective is to achieve an accuracy above 75%, demonstrating that the model can reliably predict the success of Alphabet Soup's funding applicants based on the provided features. Adjustments to the architecture, the number of epochs, and the preprocessing steps contribute to enhancing the model's accuracy and predictive power. As a result, the analysis successfully develops a binary classification neural network capable of predicting the success of funding applicants. By leveraging data preprocessing, deep learning model design, and targeted optimizations, the approach offers a powerful tool for more informed and strategic decision-making, helping Alphabet Soup maximize the impact of future funding ventures.

**Results**

**Data Preprocessing:**

* What variable(s) are the target(s) for your model? The variable target for the model is the **“IS\_SUCCESSFUL”** column, which indicates whether a funded project is successful.

A screenshot of a phone

Description automatically generated

* What variable(s) are the features for your model? The variable features for the model are the columns **“APPLICATION\_TYPE”, “AFFILICATION”, “CLASSIFICATION”, “USE\_CASE”, “ORGANIZATION”, “STATUS”, “INCOME\_AMT”, “SPECIAL\_CONSIDERATIONS”** and **“ASK\_AMT”.**

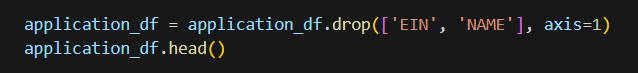
A screen shot of a black screen

Description automatically generated

* What variable(s) should be removed from the input data because they are neither targets nor features? The variables that are removed from the input data are **“EIN”** and **“NAME.”**

A black and white sign with white text

Description automatically generated

****

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

* How many neurons, layers, and activation functions did you select for your neural network model, and why? The **ReLU** activation in the first layer (with 80 neurons) helps the model understand complex patterns in the data by making it more flexible. The **sigmoid** activation in the second layer (with 30 neurons) and the output layer is used because it turns the outputs into values between 0 and 1, which is useful for deciding if something is likely to succeed or not. The first layer has more neurons (80) to pick up a lot of details from the data, while the second layer has fewer neurons (30) to focus on the most important patterns. The output layer has just one neuron to give a final prediction on whether the applicant will succeed, which is perfect for this kind of yes/no question.

A screen shot of a computer program

Description automatically generated

* Were you able to achieve the target model performance?

No, after three attempts of model optimization, 72% accuracy remained.

A black background with white text

Description automatically generated

* What steps did you take in your attempts to increase model performance?

To increase model performance, more columns were dropped from the data, such as the **“STATUS” and “SPECIAL\_CONSIDERATIONS”** columns. Also, the cutoff values for the categorical columns were reduced, and the activation functions and number of neurons were changed.

‘A screen shot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA computer screen with text on it

Description automatically generated

**Summary**

The target variable for the model is the **"IS\_SUCCESSFUL"** column, which indicates whether a funded project was successful. The feature variables used in the model are **"APPLICATION\_TYPE," "AFFILIATION," "CLASSIFICATION," "USE\_CASE," "ORGANIZATION," "STATUS," "INCOME\_AMT," "SPECIAL\_CONSIDERATIONS,"** and **"ASK\_AMT."** The **"EIN"** and **"NAME"** columns are removed since they don’t provide useful information for predicting success.

The neural network model is designed with three layers. The first hidden layer uses **ReLU** activation with 80 neurons, which helps the model understand complex patterns in the data. The second hidden layer uses **sigmoid** activation with 30 neurons to refine these patterns. The output layer also uses **sigmoid** activation and has just one neuron to provide a final prediction on whether an applicant will succeed or not. The **ReLU** function helps the model handle complex input data, while **sigmoid** is effective in binary classification tasks, where outputs range between 0 and 1, representing success or failure.

Efforts to improve the model's performance included dropping additional columns, such as **"STATUS"** and **"SPECIAL\_CONSIDERATIONS,"** reducing cutoff values for categorical variables, and experimenting with different numbers of neurons and activation functions. However, despite these adjustments, the model's accuracy remained at 72%, which fell short of the 75% target.

On the other hand, a confusion matrix is another model to evaluate how well this model is making predictions, especially for binary classification deciding if a project will succeed or fail. Instead of just looking at the model's accuracy, the confusion matrix shows **four different outcomes** of the model's predictions:

* **True Positives (TP)**: When the model correctly predicts that a project will succeed.
* **True Negatives (TN)**: When the model correctly predicts that a project will not succeed.
* **False Positives (FP)**: When the model wrongly predicts that a project will succeed, but it actually fails.
* **False Negatives (FN)**: When the model wrongly predicts that a project will fail, but it actually succeeds.

By using these four categories, the confusion matrix helps understand not only how often the model is right, but also what types of mistakes it makes (predicting success when it fails, or failure when it succeeds).