**DEEP LEARNING – REPORT ANALYSIS**

**Overview**

The non-profit foundation Alphabet Soup wants to create an algorithm to predict whether applicants for funding will be successful. A dataset of 34,000 organizations that received funding from Alphabet Soup over the years was provided to develop the algorithm. This dataset was processed through deep learning and neural networks to create a binary classifier to make the forecast.

**Results**

1. **Data Preprocessing**

The dataset contains several columns that capture the metadata of each organization:

* **EIN** and **NAME**—Identification columns
* **APPLICATION\_TYPE**—Alphabet Soup application type
* **AFFILIATION**—Affiliated sector of industry
* **CLASSIFICATION**—Government organization classification
* **USE\_CASE**—Use case for funding
* **ORGANIZATION**—Organization type
* **STATUS**—Active status
* **INCOME\_AMT**—Income classification
* **SPECIAL\_CONSIDERATIONS**—Special consideration for application
* **ASK\_AMT**—Funding amount requested
* **IS\_SUCCESSFUL**—Was the money used effectively

Graphical user interface, text

Description automatically generated Table

Description automatically generated

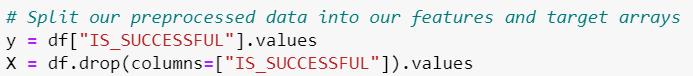
*Image #1: Overview of the dataset*

**1.1 Dropping and Manipulating Columns**

The first step of the data preprocessing was to drop all the columns that are not considered beneficial in the analysis such as EIN and NAME. Additionally, we modified some columns to have only a certain number of values. For example, we replaced any APPLICATION\_TYPE that had less than 500 counts and any CLASSIFICATION that had less than 1000 counts with the type “Other”. Lastly, we applied the pd.get\_dummies() to convert categorical data to numeric

**1.2: Separating Data into Target vs Features**

Once our data was preprocessed, we separated our data into the features and target arrays. The target array is the IS\_SUCCESSFUL column, which indicates if the money was used effectively. The rest of the columns from the processed dataset then became the features.



*Image #2: Separating Data into Target vs Features*

**1.3: Separating Training vs Testing Data and Scaling**

Then the preprocess data was divided into a training (~80%) and testing (~20%) data. Lastly, since machine learning algorithms are sensitive to large data values, the data was scaled so features are in standardized ranges.

Graphical user interface, text, application, email

Description automatically generated

*Image #3: Separating Training vs Testing Data and Scaling*

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

**2.1: Initial Modeling**

For our initial model, we created a Neural Network with 4 layers (1 input layer, 2 hidden layers and 1 output layer). The first hidden layer has 80 nodes, and the second hidden layer has 30 nodes, both with an ReLU (Rectified Linear Unit) activation function, which returns a value from 0 to infinity. Finally, the output layer used the sigmoid activation function, which transforms the output to be a range between 0 and 1.

Chart, shape, box and whisker chart

Description automatically generated with medium confidenceDiagram

Description automatically generated with low confidence

*Image #4: ReLU Activation Function (Left) vs Sigmoid Activation Function (Right)*

The model was then compiled using the “Adam” optimizer and using accuracy as the metrics. This allowed us to print the accuracy at the end of each epoch so that it is easy to judge how well the model is doing to train the data. The model was then trained with 100 epochs and the obtained loss was 0.5544 and the accuracy was 0.731.

Table

Description automatically generated

*Image #5: Initial Neural Network*

**2.2 Additional Modeling to Increase Performance**

In this new model, some of the features of the previous model were preserved. For example, the NAME column was dropped and the columns APPLICATION\_TYPE and CLASSIFICATION were manipulated to show the word “Other” if they had less than a certain number of counts.

However, some differences were introduced to increase the performance of the algorithm. When doing the value counts of the STATUS and the SPECIAL\_CONSIDERATIONS columns, it seems like over 95% of the data falls into just one category. Therefore, those 2 columns were eliminated since they do not provide meaningful information. Lastly, in this case, the NAME column was preserved and only those applications with names that appear less than 5 times were eliminated.

Once our data was preprocessed, we separated our data into the target arrays (column IS\_SUCCESSFUL) and the features (rest of the column) and the data was scaled.

For the model, we created a Neural Network with 5 layers (1 input layer, 3 hidden layers and 1 output layer). The first hidden layer has 100 nodes, the second hidden layer has 50 nodes, and the third hidden layer had 10 nodes, all of them using the ReLU (Rectified Linear Unit) activation function.

Table

Description automatically generated

*Image #6: Improved Neural Network*

The model was then compiled using the “Adam” optimizer and it was trained using 300 epochs. The obtained loss was 0.5189 and the accuracy was 0.7924.

**Summary**

1. **Summary of the Results**

The goal Alphabet Soup is to create an algorithm to predict whether applicants for funding will be successful that is at least 75% accurate. The initial model, which consisted of 4 layers of neural networks did not achieve the 75% goal. However, when preprocessing the data slightly differently (not dropping the NAME column and dropping STATUS and SPECIAL CONSIDERATIONS columns) and creating a neural network with 5 layers and more nodes, an accuracy of 79% was obtained.

1. **Recommendations for Different Models**

As a comparison, a Random Forest classifier could also be used to solve this classification problem When tested and an accuracy of 77.9% was obtained. Even though it did not perform better than the improved neural network (79% accuracy), it performed better that the initial neural network (73% accuracy).