**Neural Network Model Report**

**Objective**

Develop a tool for the nonprofit foundation Alphabet Soup to help select applicants for funding with the best chance of success in their ventures. Apply machine learning and neural network modeling using the features within the given dataset, containing more than 34,000 organizations that have received funding from Alphabet Soup over the years, to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

**Results**

**Data Preprocessing**

**Step 1**: Identify variables and categorize.

* What variable(s) are the target(s) for your model?
* IS\_SUCCESSFUL—Was the money used effectively
* What variable(s) are the features for your model?
* 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 considerations for application
* ASK\_AMT—Funding amount requested
* What variable(s) should be removed from the input data because they are neither targets nor features?
* EIN and NAME—Identification columns

A close-up of a name

Description automatically generated

**Step 2:** Identify unique column values and encode to apply a categorical value to create an all numeric dataset.

A screenshot of a computer code

Description automatically generated

A close-up of a computer code

Description automatically generated

A computer screen shot of a computer code

Description automatically generated**Step 3:** Reduce the number of categorical values with few unique values (aka: bucketing or binning).

**Step 4:** Split input data into training and test datasets.

A computer code with text

Description automatically generated with medium confidence**Step 5:** Standardize the numerical data using scikit-learn StanardScaler module to reduce the overall likelihood that outliers or skewed distributions will have a negative impact on the model’s performance.

A close-up of a white background

Description automatically generated

**Step 6:** Fit StandardScalar on training features.

A screenshot of a computer

Description automatically generated

**Step 7:** Scale training and test feature set.

A close-up of a computer code

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 was initially chosen because the data is not linear and ReLU is known to be the most popular activation function for deep learning models as it offers advantages such as computational power for classification data. Throughout the optimization process activations were changed, layers and neurons were added, and epochs were increased.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Neurons** | **Layers** | **Activation Functions** | **Epochs** | **Accuracy** |
| **Original Model** | 6, 6 | 2 hidden, 1 output | Hidden: ReLU, ReLU  Output: Sigmoid | 100 | 73.00% |
| **Model 1** | 6, 6 | 2 hidden, 1 output | Hidden: ReLU, ReLU  Output: ReLU | 100 | 73.18% |
| **Model 2** | 6, 6, 6 | 3 hidden, 1 output | Hidden: ReLU, ReLU, ReLU Output: ReLU | 100 | 72.98% |
| **Model 3** | 6, 6 | 2 hidden, 1 output | Hidden: Leaky ReLU, Leaky ReLU  Output: Sigmoid | 100 | 73.10% |
| **Model 4** | 6, 6, 6 | 3 hidden, 1 output | Hidden: Leaky ReLU, Leaky ReLU, Sigmoid  Output: Sigmoid | 100 | 73.08% |
| **Model 5** | 6, 6, 6, 6 | 4 hidden, 1 output | Hidden: Leaky ReLU, Leaky ReLU, Sigmoid, Sigmoid  Output: Sigmoid | 100 | 73.05% |
| **Model 6** | 12, 12, 12 | 3 hidden, 1 output | Hidden: Leaky ReLU, Leaky ReLU, Sigmoid  Output: Sigmoid | 100 | 72.93% |
| **Model 7** | 6, 6, 6 | 3 hidden, 1 output | Hidden: Leaky ReLU, Leaky ReLU, Sigmoid  Output: Sigmoid | 200 | 72.82% |

* Were you able to achieve the target model performance?

No, after several neural networking model attempts the highest accuracy achieved was 73.18% in model 2.

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

**Model Optimization Techniques**

* Added an additional layers
* Added additional neurons
* Increased epochs

**Summary**

Analyzing the several models developed and ran, the best model for Alphabet Soup to predict whether applicants will be successful if funded is model 3. Though model 3 doesn’t produce the highest accuracy of 73.18%, it is a close second at 73.10%. The .08% accuracy difference is made up for under “loss”. Model 3 produces one of the lowest loss percentages at 55.09%, with a difference of 2.5% compared to model 2.

**Model 2**

A screenshot of a computer code

Description automatically generated

**Model 3**

A computer code with text

Description automatically generated with medium confidence

As a suggested alternative model, the random forest model is a great choice for Alphabet Soup because it is resistant to outliers and can evaluate the importance of a specific feature in the data. The model will also cost less and still be able to handle complex problems as it works with classification data.