**Neural Network Charity Analysis**

This module was an exercise in Neural Networks and Deep Learning Models using the following:

- TensorFlow and Pandas libraries in Python to preprocess datasets and create a predictive binary classifier.

**Module Overview**

The purpose of this analysis was to explore and implement neural networks using TensorFlow in Python. Neural networks machine learning algorithms can recognize features and patterns in ta dataset. Modeled after the human brain neural networks are modeled contain neurons in layers that perform individual computations.

In this module, we learned:

- How to create a basic neural network

- Preprocess and prepare the datasets

- Build a training and testing set

- Measure the model accuracy

- Add additional neurons and hidden layers to optimize the model

- How to select a best model to use for our dataset

**AlphabetSoup** isrequesting a mathematical data solution that will help determine which organizations merit donating to and which ones are considered to high-risk. Not every donation AlphabetSoup has made to an organization has been successful.

**The** task is to analyze the impact of each AlphabetSoup organization that they donated to in order to vet them and determine if the AlphabetSoup's money will be used successfully. To do this AlphabetSoup’s needs a binary classifier that will predict if an organization uses the donation successfully.

## Preprocessing the Data for a Neural Network

A CSV file “charity\_data.csv” containing more than 34,000 organizations that have received past contributions. The following information was contained within this dataset.

CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. The data was preprocessed as follows:

**- EIN** and **NAME**columns were removed during the preprocessing stage as these columns added no value.

- The **APPLICATION\_TYPE** column data was binned and any unique values with less than 500 records as were categorized as "Other"

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

The following parameters were used to **compile, train, and evaluate the model.** The initial model had a total of 5,981 parameters as a result of 43 inputs with 2 hidden layers and 1 output layer.

- The first hidden layer had 43 inputs, 80 neurons.

- The second hidden layer had 80 inputs, 30 neurons.

- The output layer had 30 inputs, 1 neuron.

- Both the first and second hidden layers were activated using the Rectified Linear Unit function (relu). The output layer was activated using the Sigmoid function.

An accuracy rate of greater than 75% was the target performance rate. This model only achieved an accuracy rate of 73.73%

image

### **Attempts to Optimize and Improve the Accuracy Rate**

Three additional attempts were made to increase the performance of the model by changing by adding/subtracting neurons and epochs. No improvement was seen in the three attempts.

image

**Optimization 1**

- Binned **INCOME\_AMT** column

- Created 5,821 total parameters, a decrease from the original of 5,981

- Accuracy decreased 0.78% from 73.73%% to 72.95%

- Loss was reduced by 0.10% from 56.33% to 56.23%

image

**Optimization 2:**

Removed the “**ORGANIZATION”** column

Binned “**INCOME\_AMT”**column

Removed “**SPECIAL\_CONSIDERATIONS\_Y”** column from features as it is redundant to “**SPECIAL\_CONSIDERATIONS\_N”**

Increased neurons to 200 for the first hidden layer and 100 for the second hidden layer

Created 28,001 total parameters, an increase from the original of 5,981

Accuracy decreased 0.11% from 72.33% to 72.65%

Loss increased by 1.75% from 58.08% to 58.02%

image

**Optimization 3:**

Binned “**INCOME\_AMT”** and “**AFFILIATION”** column

Removed the “**ORGANIZATION”** column

Removed “**SPECIAL\_CONSIDERATIONS\_Y”** column from features as it is redundant to “**SPECIAL\_CONSIDERATIONS\_N”**

Increased neurons to 400 for the first hidden layer and 200 for the second hidden layer

* + Created 92,801 total parameters, an increase from the original of 5,981
  + Accuracy increased 0.19% from 72.33% to 72.33%
  + Loss decreased by 1.82% from 58.08% to 57.51%

image

## Summary

In summary, the model and the three optimizations did not achieve the desired result of greater than 75%.