Deep Learning with AlphabetSoup Analysis

Overview:

The primary goal of this analysis is to build and optimize a deep learning model to predict the success of funding applicants for the nonprofit foundation Alphabet Soup. Given a dataset of over 34,000 organizations that have previously received funding, we aim to develop a binary classifier that can accurately determine whether future applicants will use the funding effectively. Success in this context is indicated by the IS_SUCCESSFUL target variable.

Data Processing:

Target Variable(s) for the Model:

- **IS_SUCESSFUL** - Was the money used effectively

Feature Variable(s) for the 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

Removed Data:

- NAME and EIN - these are identification columns

Modeling:

Model layout: I started with a one hidden layer model utilizing 5 nodes and 250 epochs. I utilized only one activation type as there was only one hidden layer.

Compile, Train and Evaluate the Model

[27]:
```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE

nn_model = tf.keras.models.Sequential()

# Add our first Dense layer, including the input layer
nn_model.add(tf.keras.layers.Dense(units=5, activation="tanh", input_dim=len(X.columns)))

# Add the output layer that uses a probability activation function
nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the Sequential model
nn_model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 5)	230
dense_10 (Dense)	(None, 1)	6

Total params: 236 (944.00 B)
Trainable params: 236 (944.00 B)
Non-trainable params: 0 (0.00 B)

Were you able to achieve target model performance? No I was not

Steps taken to increase performance: To improve performance I attempted to add layers and neurons at these layers, different types of activations, and number of epochs

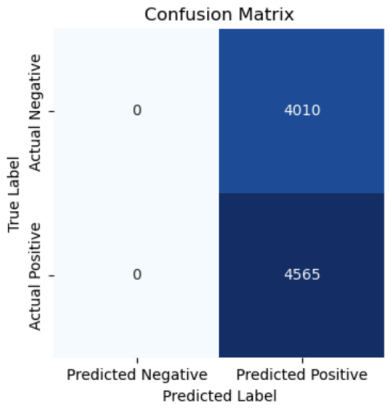
Summary:

This model and optimization model are both poor models to attempt to use to predict whether or not AlphabetSoup's investment into applicants will be successful or not. The initial model only predicts positives and the optimized model only predicts negatives. This modeling could be improved by changing the inputs during the data pre-processing. One example could be by binning and utilizing the name column. As well, attempting other types of predictive modeling could be more successful ie. linear regression or XGBoost as XGBoost does well when predicting and working with tabular data.

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      4010
           1       0.53      1.00      0.69      4565

    accuracy                           0.53      8575
   macro avg       0.27      0.50      0.35      8575
weighted avg       0.28      0.53      0.37      8575
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.47      1.00      0.64      4010
           1       0.00      0.00      0.00      4565

    accuracy                           0.47      8575
   macro avg       0.23      0.50      0.32      8575
weighted avg       0.22      0.47      0.30      8575
```