# Deep Learning Project: Charity Funding Predictor

Deep learning and neural networks were used to determine if applicants would be successful funded by Alphabet Soup, Whom previously funded by 34,000 organizations.

## Data Processing

Using your knowledge of Pandas and scikit-learns `StandardScaler()`, preprocessed the dataset. This step prepares for Step 2, where we compile, train, and evaluate the neural network model.

Using the information in the Challenge files, followed the instructions to complete the preprocessing steps.

1. Read in the `charity_data.csv` to a Pandas DataFrame , identified

- What variable(s) are the target(s) for the model
- What variable(s) are the feature(s) for the model

2. Dropped the `EIN` and `NAME` columns. The remaining columns were considered features for the model.
3. Determine the number of unique values for each column.
4. For columns that have more than 10 unique values, determined the number of data points for each unique value.
5. Using the number of data points for each unique value to pick a cutoff point to bin "rare" categorical variables together in a new value, `Other`, and then checked the binning was successful or not.
6. Using `pd.get_dummies()` to encode categorical variables.
7. Splitted the preprocessed data into a features array, `X`, and a target array, `y`. Using these arrays and the `train_test_split` function to split the data into training and testing datasets.
8. Scaled the training and testing features datasets by creating a `StandardScaler` instance, fitting it to the training data, then using the `transform` function.

## Compile, Train, and Evaluate the Model

Neural network was applied to each model multiple layers, three in total. The number of features decided the number of hidden nodes.

# Deep Learning Project: Charity Funding Predictor

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len( X_train_scaled[0])
hidden_nodes_layer1=7
hidden_nodes_layer2=14
hidden_nodes_layer3=21
nn = tf.keras.models.Sequential()

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 378

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 505 (1.97 KB)
Trainable params: 505 (1.97 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5550 - accuracy: 0.7272 - 205ms/epoch - 765us/step
Loss: 0.5549846291542053, Accuracy: 0.7272303104400635
```

A 3 layer  training generated 505 parameters. The first attempt came close at 72% which is under the desired 75%.

## Optimization

# Deep Learning Project: Charity Funding Predictor

The second attempt added “NAME” back into the dataset, this time I achieved 78% which was 3% over target.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 3171

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 3298 (12.88 KB)
Trainable params: 3298 (12.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 1s - loss: 0.4677 - accuracy: 0.7896 - 584ms/epoch - 2ms/step
Loss: 0.4677150547504425, Accuracy: 0.7896209955215454
```