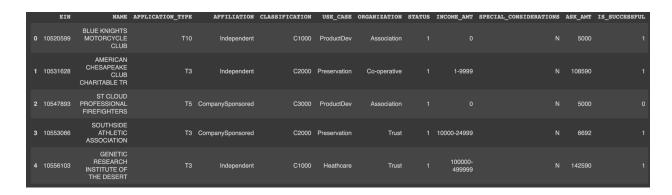
Analysis

The nonprofit foundation Alphabet Soup needed help selecting applicants that would have the best chance of success in their ventures. We used machine learning and neural networks to create a tool that can help predict which applicants would be successful out of a table of 34,000 organizations.



Results

Data Preprocessing

- First, we eliminate columns that do not provide any benefits, such as "EIN" and "NAME."
- The remaining columns are considered as the features used in the model.
- Next, we divide the data into separate sets for testing and training the model.
- The target variable is denoted as "IS_SUCCESSFUL" with a value of 1 indicating "yes" and 0 indicating "no."
- The application data is analyzed while the classification data is used for binning.
- Then, we encoded the categorical variables with get dummies().

Compiling, Training, and Evaluating the Model

- In order to determine the optimal number of neurons, layers, and activation functions, we employed Keras Tuner.
- For our neural network models, we utilized three layers.
- The number of hidden nodes was determined by the number of features.

```
# 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()
    # First hidden laver
   nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))
   nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))
    # Output layer
   nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
Model: "sequential 1"
    Layer (type)
                             Output Shape
                                                      Param #
    dense_3 (Dense)
                              (None, 7)
    dense_4 (Dense)
                              (None, 14)
    dense_5 (Dense)
                               (None, 1)
   Total params: 442
    Trainable params: 442
   Non-trainable params: 0
```

 By implementing these three layers, we obtained 442 parameters, resulting in an accuracy of 72%.

```
[47] # 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: 0.5536 - accuracy: 0.7226 - 348ms/epoch - lms/step
Loss: 0.5535905361175537, Accuracy: 0.7225655913352966
```

Optimization and Summary

- To optimize results and attempt to reach at least 75% accuracy, we tried only removing the "EIN" column and keeping the "NAME" column.
- By doing this, we reached 841 params and 76% accuracy.
- In summary, keeping the "NAME" column proved to be beneficial in optimizing the model to improve performance.

```
[44] # 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: 0.4843 - accuracy: 0.7643 - 436ms/epoch - 2ms/step
    Loss: 0.4842992126941681, Accuracy: 0.7643148899078369
```