Deep Learning Project: Charity Funding Predictor

In this project we will use the artificial intelligence technologies: Deep learning and neural networks, in order to determine if some applicants would be successfully funded by Alphabet Soup, whom previously funded over 34,000 organizations.

Step 1: Data Processing

The dataset that we worked on is a .csv file. We first removed any irrelevant information; therefore, we dropped EIN and NAME columns from the dataset. The remaining columns were considered features for the model. CLASSIFICATION and APPLICATION_TYPE columns were replaced with 'Other' due to high fluctuation. After that the data was split into training and testing sets of data with 33 percent to testing data. Our target variable for the model is "IS_SUCCESSFUL" column, 1 is considered yes and 0 means no for the application. APPLICATION data was analyzed, and CLASSIFICATION's value was used for binning. Each unique value used several data point as a cutoff point to bin "rare" categorical variables together in a new value, 'Other'. Afterwards checked to see if binning was successful. Categorical variables were encoded by 'pd.get_dummies().

Step 2: Compiling, Training, and Evaluation the Model

Neural Network was applied on each model multiple layers, two hidden layers and one output layers, which means 3 in total. The number of features dictated the number of hidden nodes.

```
# 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

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)                 350
_________________________________________________________________
dense_1 (Dense)              (None, 14)                112
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 15
=================================================================
Total params: 477
Trainable params: 477
Non-trainable params: 0
```

A three-layer training model generated 477 parameters. The first attempt came close at 72% which wasunder our objective: 75%.

```
# 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}")
```

```
354/354 - 0s - loss: 0.5531 - accuracy: 0.7257
Loss: 0.5531102418899536, Accuracy: 0.7256824970245361
```

Step 3: Optimize the Mode

In order to optimize our model, we added 'NAME' column back into the dataset, this time we achieved 77% with a total of 2,038 params.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 7)                 1911
_________________________________________________________________
dense_1 (Dense)              (None, 14)                112
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 15
=================================================================
Total params: 2,038
Trainable params: 2,038
Non-trainable params: 0
_________________________________________________________________
```

```
# 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}")
```

```
354/354 - 0s - loss: 0.4622 - accuracy: 0.7775
Loss: 0.46216529607772827, Accuracy: 0.7775421738624573
```

Deep learning models should have multiple layers, since it is machined based, it teaches a computer to filter inputs through the layers to learn how to predict and classify information