Jonathan Martinez – Data Analyst

Alphabet Soup Charity – Neural Network Report

1. Overview of the analysis: Explain the purpose of this analysis.

The objective of conducting the charity funding analysis for Alphabet Soup was to forecast the company's investment approvals. The aim was to leverage machine learning and neural networks to employ target and features on the dataset, thereby developing a binary classifier capable of predicting the likelihood of success for investors funded by Alphabet Soup. The analysis commenced with a dataset comprising 34,000 organizations and 12 columns encompassing metadata related to each organization and their historical funding outcomes.

2. Results:

INITIAL MODEL –

Quick Result: Loss: 0.5549299716949463, Accuracy: 0.7289795875549316
TARGET NOT ACHIEVED

Data Processing:

2.1. What variable(s) are the target(s) for your model?

Answer: IS_SUCCESSFUL is the target that is marked 1 for successful.

2.2. What variable(s) are the features for your model?

Answer: Since identification columns 'EIN' and 'NAME' were removed.
The feature variables of the model are:

- APPLICATION_TYPE,
- AFFILIATION,
- CLASSIFICATION,
- USE_CASE,
- ORGANIZATION,
- STATUS,
- INCOME_AMT,
- SPECIAL_CONSIDERATIONS,
- ASK_AMT.

2.3. What variable(s) should be removed from the input data because they are neither targets nor features?

Answer: EIN and NAME were removed from the input data because they are neither targets nor features.

Compiling, Training, and Evaluating the Model:

The following pictures answer the questions about number of neurons used, layers and activation functions. It also displays the target model performance.

```
# 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 = 10
hidden_nodes_layer2 = 5

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, 10)                750

 dense_1 (Dense)             (None, 5)                 55

 dense_2 (Dense)             (None, 1)                 6

=================================================================
Total params: 811
Trainable params: 811
Non-trainable params: 0
_________________________________________________________________
```

```
# Train the model
fit_model_1 = nn.fit(X_train_scaled, y_train, epochs=50,callbacks=[callback])
```

```
# 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: 0.5549 - accuracy: 0.7290 - 341ms/epoch - 1ms/step
Loss: 0.5549299716949463, Accuracy: 0.7289795875549316
```

FIRST OPTMISATION MODEL

Quick Result: Loss: 0.4941253066062927, Accuracy: 0.7514868974685669
TARGET ACHIEVED.

Data Processing:

2.4. What variable(s) are the target(s) for your model?
Answer: IS_SUCCESSFUL is the target that is marked 1 for successful.

2.5. What variable(s) are the features for your model?
Answer: Since identification columns 'EIN' were removed.
The feature variables of the model are:

- NAME,
- APPLICATION_TYPE,
- AFFILIATION,
- CLASSIFICATION,
- USE_CASE,
- ORGANIZATION,
- STATUS,
- INCOME_AMT,
- SPECIAL_CONSIDERATIONS,
- ASK_AMT.

2.6. What variable(s) should be removed from the input data because they are neither targets nor features?
Answer: EIN were removed from the input data because they are neither targets nor features.

Compiling, Training, and Evaluating the Model:

The following pictures answer the questions about number of neurons used, layers and activation functions. It also displays the target model performance.

Changes compared to initial model:

- Nulls and duplicated were validated.
- Included the column NAME as they could be grouped/binned in a smaller size and help to improve the model.

```
# 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 = 10
hidden_nodes_layer2 = 5

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, 10)                750

 dense_1 (Dense)             (None, 5)                 55

 dense_2 (Dense)             (None, 1)                 6

=================================================================
Total params: 811
Trainable params: 811
Non-trainable params: 0
_________________________________________________________________
```

```
# Train the model
fit_model_1 = nn.fit(X_train_scaled, y_train, epochs=50,callbacks=[callback])
```

```
# 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: 0.4941 - accuracy: 0.7515 - 634ms/epoch - 2ms/step
Loss: 0.4941253066062927, Accuracy: 0.7514868974685669
```

SECOND OPTMISATION MODEL

Quick Result: Loss: 0.4942009747028351, Accuracy: 0.7601166367530823
TARGET ACHIEVED AND IMPROVED.

Data Processing:

2.7. What variable(s) are the target(s) for your model?
Answer: IS_SUCCESSFUL is the target that is marked 1 for successful.

2.8. What variable(s) are the features for your model?
Answer: Since identification columns 'EIN' were removed.
The feature variables of the model are:

- NAME,
- APPLICATION_TYPE,
- AFFILIATION,
- CLASSIFICATION,
- USE_CASE,
- ORGANIZATION,
- STATUS,
- INCOME_AMT,
- SPECIAL_CONSIDERATIONS,
- ASK_AMT.

2.9. What variable(s) should be removed from the input data because they are neither targets nor features?
Answer: EIN were removed from the input data because they are neither targets nor features.

Compiling, Training, and Evaluating the Model:

The following pictures answer the questions about number of neurons used, layers and activation functions. It also displays the target model performance.

Changes compared to initial model:

- Nulls and duplicated were validated.
- Included the column NAME as they could be grouped/binned in a smaller size and help to improve the model.
- Increased the number of nodes in each layer and the epochs.

```
# 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 = 20
hidden_nodes_layer2 = 10

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, 20)                1500

 dense_1 (Dense)             (None, 10)                210

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 1,721
Trainable params: 1,721
Non-trainable params: 0
_________________________________________________________________
```

```
# Train the model
fit_model_1 = nn.fit(X_train_scaled, y_train, epochs=70,callbacks=[callback])
```

```
# 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: 0.4942 - accuracy: 0.7601 - 388ms/epoch - 1ms/step
Loss: 0.4942009747028351, Accuracy: 0.7601166367530823
```

THIRD OPTMISATION MODEL

Quick Result: Loss: 0.49886831641197205, Accuracy: 0.7573177814483643
TARGET ACHIEVED.

Data Processing:

2.10. What variable(s) are the target(s) for your model?
Answer: IS_SUCCESSFUL is the target that is marked 1 for successful.

2.11. What variable(s) are the features for your model?
Answer: Since identification columns 'EIN' were removed.
The feature variables of the model are:
- NAME,
- APPLICATION_TYPE,
- AFFILIATION,
- CLASSIFICATION,
- USE_CASE,
- ORGANIZATION,
- STATUS,
- INCOME_AMT,
- SPECIAL_CONSIDERATIONS,
- ASK_AMT.

2.12. What variable(s) should be removed from the input data because they are neither targets nor features?
Answer: EIN were removed from the input data because they are neither targets nor features.

Compiling, Training, and Evaluating the Model:

The following pictures answer the questions about number of neurons used, layers and activation functions. It also displays the target model performance.

Changes compared to initial model:

- Nulls and duplicated were validated.
- Included the column NAME as they could be grouped/binned in a smaller size and help to improve the model.
- Increased the number of nodes in each layer and the epochs**.**

```
# 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 =30
hidden_nodes_layer2 = 15
hidden_nodes_layer3 = 5

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"))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation="relu"))

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

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 30)                2250

 dense_5 (Dense)             (None, 15)                465

 dense_6 (Dense)             (None, 5)                 80

 dense_7 (Dense)             (None, 1)                 6

=================================================================
Total params: 2,801
Trainable params: 2,801
Non-trainable params: 0
_________________________________________________________________
```

```
# Train the model
fit_model_1 = nn.fit(X_train_scaled, y_train, epochs=80,callbacks=[callback])
```

```
# 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: 0.4989 - accuracy: 0.7573 - 499ms/epoch - 2ms/step
Loss: 0.49886831641197205, Accuracy: 0.7573177814483643
```

3. **Summary.**

The second neural network optimised model provides the better accuracy of 76% compared to 72% of the initial model. Then, it seems that by increasing the neurons on each layer would perform better than adding another layer. Also, the number of epochs between 70 and 80 would provide a better result as well.

Jonathan Martinez – Data Analyst