Data Processing:

In the original model the data removed any irrelevant information. This included the EIN and the name. However in the optimized model the name was added back in. Classification and application_type were replaced before splitting the data. The data was split into two sets of train and test data. The target variable was “IS_Successful” and is verified by the value, 1 was considered yes and 0 was no. Each unique value used several data points as a cutoff point to bin the “rare” categorigcal variables together in a new value. I finished up the processing by encoding the categorial values with dummies.

Compiling, Training and Evaluation of the Model:

For the first model I used a total of 2 layers. This model generated 463 Params and was 73% accurate. This was a little low for my liking.

```
# 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()
```

```
268/268 - 0s - loss: 0.5515 - accuracy: 0.7332 - 82ms/epoch - 304us/step
Loss: 0.5515455007553101, Accuracy: 0.7331778407096863
```

Optimization:

On my second model I chose to add the “Name” back into the data. This was highly effective as I achieved about 79% accuracy with over 4200 params. I also added an additional layer to the model so now there was 3 in total.

```
268/268 - 0s - loss: 0.4475 - accuracy: 0.7890 - 93ms/epoch - 347us/step
Loss: 0.44754186272621155, Accuracy: 0.7890378832817078
```

```python
# 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=8
hidden_nodes_layer2=16
hidden_nodes_layer3=26

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='sigmoid'))

# 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, 8)                 3624

 dense_1 (Dense)             (None, 16)                144

 dense_2 (Dense)             (None, 26)                442

 dense_3 (Dense)             (None, 1)                 27

=================================================================
Total params: 4237 (16.55 KB)
Trainable params: 4237 (16.55 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```