The Neural Network Model Report

Overview:

The non-profit organization Alphabet Soup aims to develop an algorithm for predicting the success of funding applicants. Leveraging the principles of machine learning and neural networks, the goal is to utilize the dataset's features to construct a binary classifier capable of determining whether applicants funded by Alphabet Soup will achieve success or not.

Results:

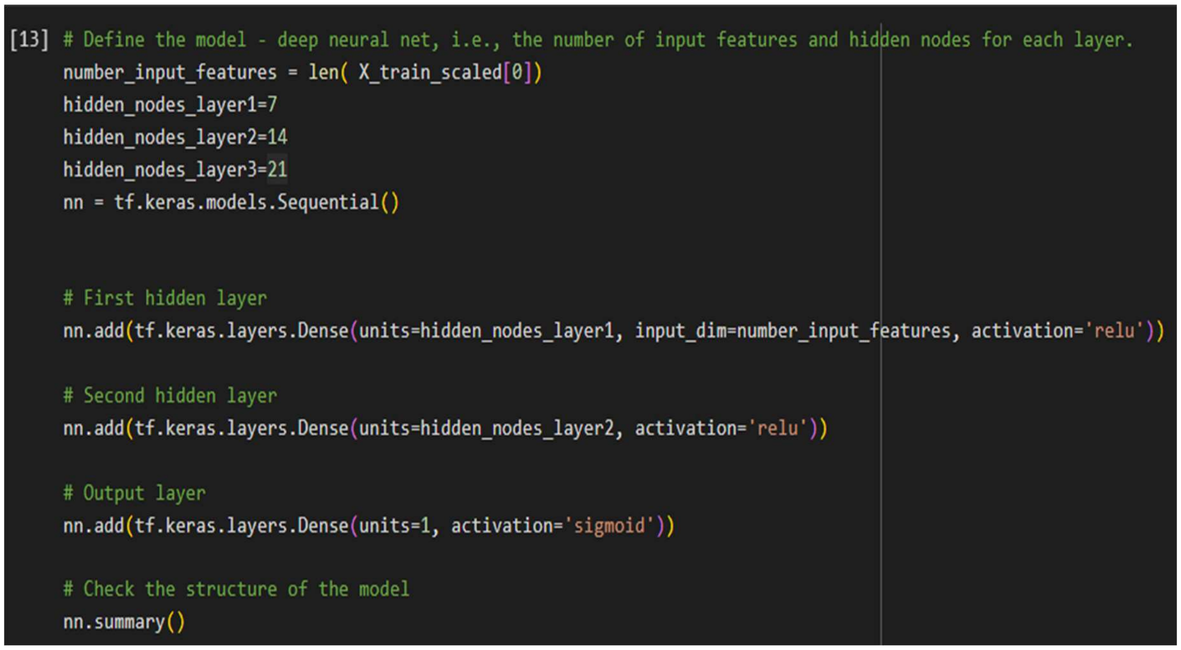
- ### Data Preprocessing:

To initiate the data processing, we began by eliminating any irrelevant information. After removing both the EIN and NAME columns, the remaining columns were designated as features for the model. The NAME column was reintroduced in the second round of testing for the purpose of binning. Subsequently, the data was divided into training and testing sets. The target variable for the model was labeled "IS_SUCCESSFUL" and was assigned a value of 1 for "yes" and 0 for "no."

For the analysis of the APPLICATION data, the "CLASSIFICATION" variable was employed for binning purposes. We established certain data thresholds to group together "rare" variables and assigned a new label of "Other" for each unique value below the threshold. After verifying the success of the binning process, categorical variables were encoded using the get_dummies() method.

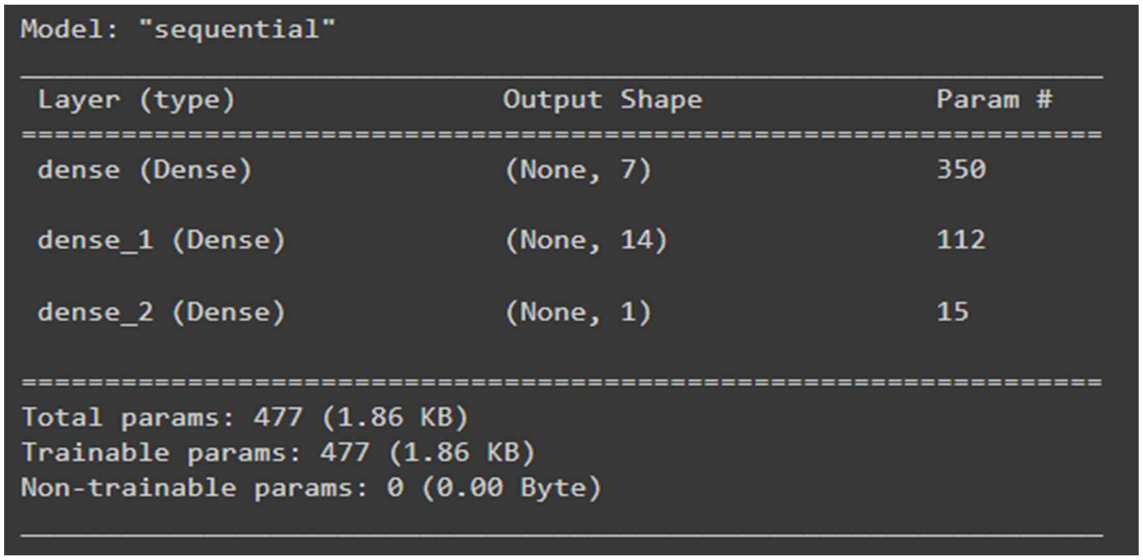
- ### Compiling, Training, and Evaluating the Model:

In each neural network model, there were a total of three layers. The number of hidden nodes in these layers was determined based on the number of features in the dataset.

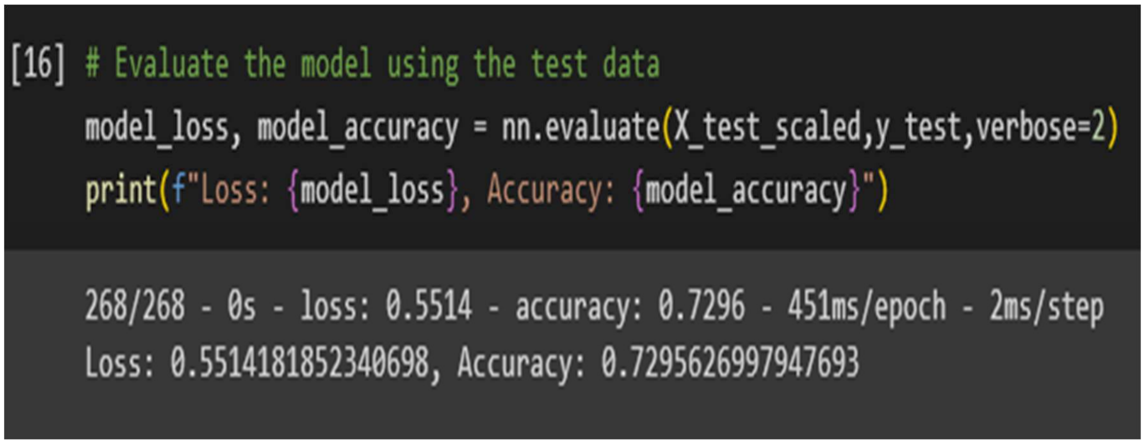


```
[13] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
     number_input_features = len( X_train_scaled[0])
     hidden_nodes_layer1=7
     hidden_nodes_layer2=14
     hidden_nodes_layer3=21
     nn = tf.keras.models.Sequential()


     # First hidden 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()
```

The three-layer training model created a total of 477 parameters. In the first attempt, it achieved an accuracy rate of just over 73%, which, while falling slightly below the desired threshold of 75%, still demonstrates promising performance and is not too far from the target accuracy.



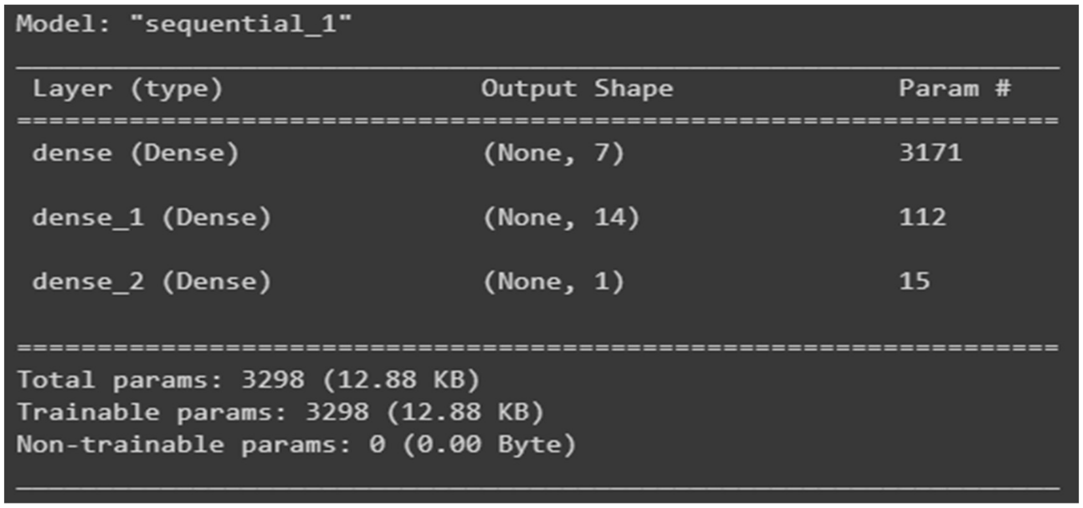
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 350

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 477 (1.86 KB)
Trainable params: 477 (1.86 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



```
[16] # 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: 0.5514 - accuracy: 0.7296 - 451ms/epoch - 2ms/step
     Loss: 0.5514181852340698, Accuracy: 0.7295626997947693
```

Optimizing The Model:

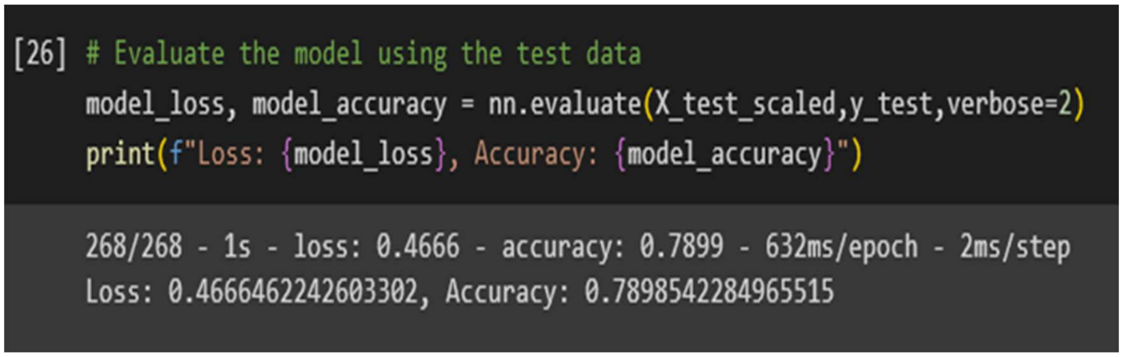
In the second attempt, the model achieved an accuracy of approximately 79%. This performance exceeded the desired accuracy target of 75% by 4%, demonstrating improved results. The model employed a total of 3,298 parameters.



```
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)
_________________________________________________________________
```

Indeed, the use of multiple layers is a fundamental characteristic of deep learning models. Deep learning leverages neural networks with multiple hidden layers to learn and extract hierarchical representations from data. As information passes through these layers, the model can automatically filter and transform the input data, enabling it to learn complex patterns and features that might not be easily discernible with a single-layer model.

This hierarchical feature extraction allows deep learning models to excel in tasks such as image recognition, natural language processing, and many other complex problems where the data has intricate structures. Deep learning models are capable of automatically learning relevant features from raw data, making them highly effective in various domains.



```
[26] # 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: 0.4666 - accuracy: 0.7899 - 632ms/epoch - 2ms/step
     Loss: 0.4666462242603302, Accuracy: 0.7898542284965515
```