Module 21: Deep Learning Challenge

Alphabet Soup Venture Funding Analysis

Caleb Meinke

Data Analytics Bootcamp, 03/25/25

Project Overview and Objectives:

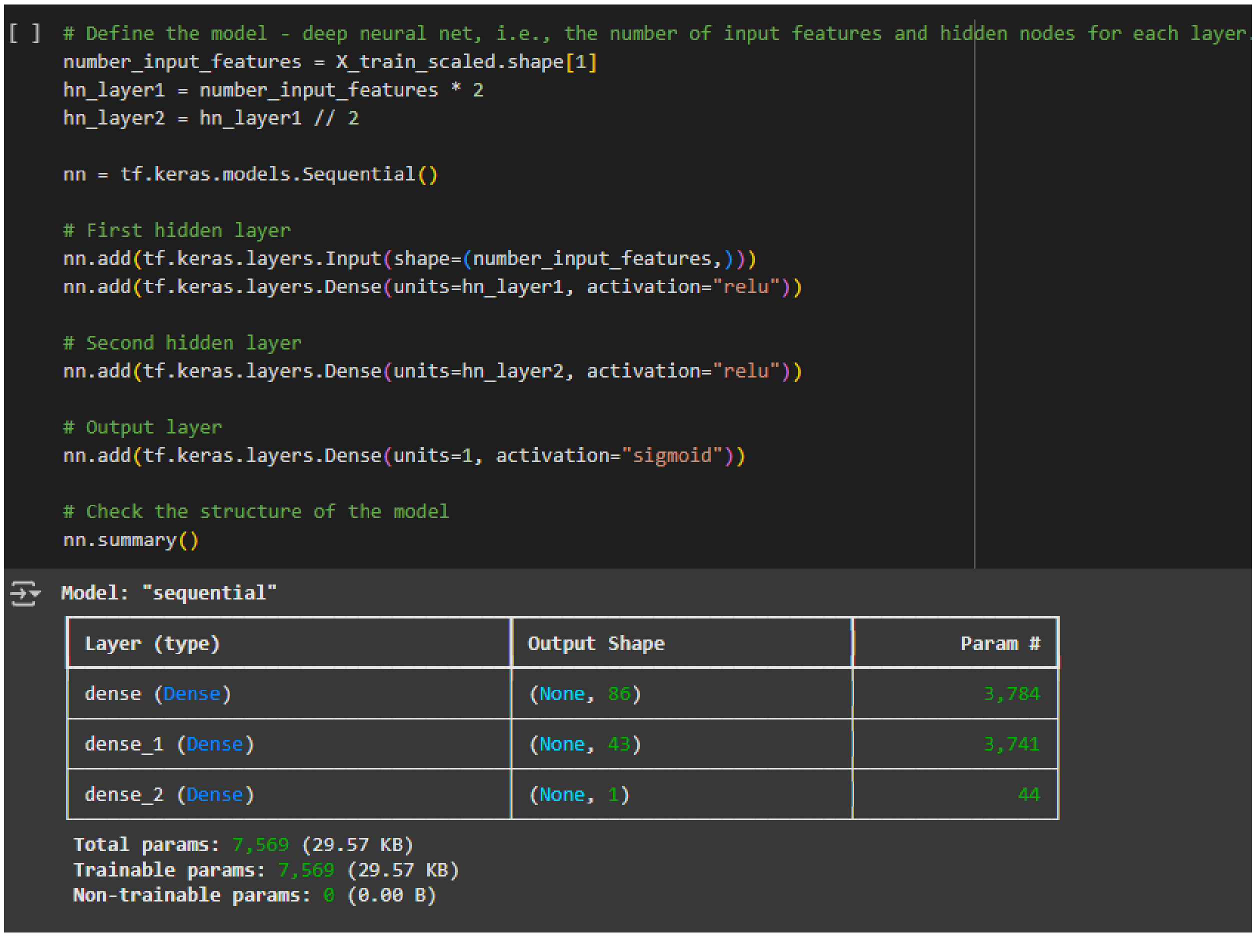
The purpose of this analysis is to develop a neural network capable of assessing and classifying funding applicants for the nonprofit foundation, Alphabet Soup. Objectives of this effort include creating a model that predicts the binary output of success of failure for each applicant with 75% accuracy or greater. The dataset used in the analysis includes metadata for the more than 34,000 organizations that have received funding from Alphabet Soup since the organization’s inception, which provides the basis for neural network development, testing and implementation.

Data Preprocessing and Model Development:

Initial focus of this analysis leveraged basic data cleaning to preprocess and organize the Alphabet Soup metadata into a workable structure. At first, this required dropping some data deemed unessential to the analysis, including the “Name” and “EIN” columns stored in the data frame. From there, the analysis focused on two target features in the data that had higher quantity unique attributes, while also permitting the data to be segmented into organized classifications. These features included the “application type” and “classification” for each organization in the metadata, which was subsequently converted from categorical data to numeric data via One Hot Encoding for model ingestion and testing under the model objectives.

Model Compilation, Training and Testing:

Development of the Alphabet Soup applicant model first uses Train-Test-Split methodology to define the training and test populations using an 80% to 20% split, with a random state of the customary 42 random states used as a default in many code structures. Thereafter, a Standard Scaler is applied to normalize and standardize the data and improve model accuracy. Model compilation during this initial phase included input of the selected features, with two hidden layers reflecting 2x the initial inputs and ½ the initial outputs, respectively. Using Keras, the first iteration of the model applies Relu and Sigmoid functions to each layer, which produces 7,569 trainable and 0 non-trainable parameters.



```
[ ] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
    number_input_features = X_train_scaled.shape[1]
    hn_layer1 = number_input_features * 2
    hn_layer2 = hn_layer1 // 2

    nn = tf.keras.models.Sequential()

    # First hidden layer
    nn.add(tf.keras.layers.Input(shape=(number_input_features,)))
    nn.add(tf.keras.layers.Dense(units=hn_layer1, activation="relu"))

    # Second hidden layer
    nn.add(tf.keras.layers.Dense(units=hn_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, 86)	3,784
dense_1 (Dense)	(None, 43)	3,741
dense_2 (Dense)	(None, 1)	44

```
Total params: 7,569 (29.57 KB)
Trainable params: 7,569 (29.57 KB)
Non-trainable params: 0 (0.00 B)
```

With the model structure complete, compilation of the date uses the "Adam" optimizer and binary cross entropy to reduce incorrect data class labeling in the model. A callback function establishes a cadence of five epochs between saved checkpoints, and execution of model training using 100 epochs produces the first round of testing results. Analysis of the output showed an initial model accuracy of 72.78%, which fell below the model objectives and required further optimization.

Model Optimization Efforts:

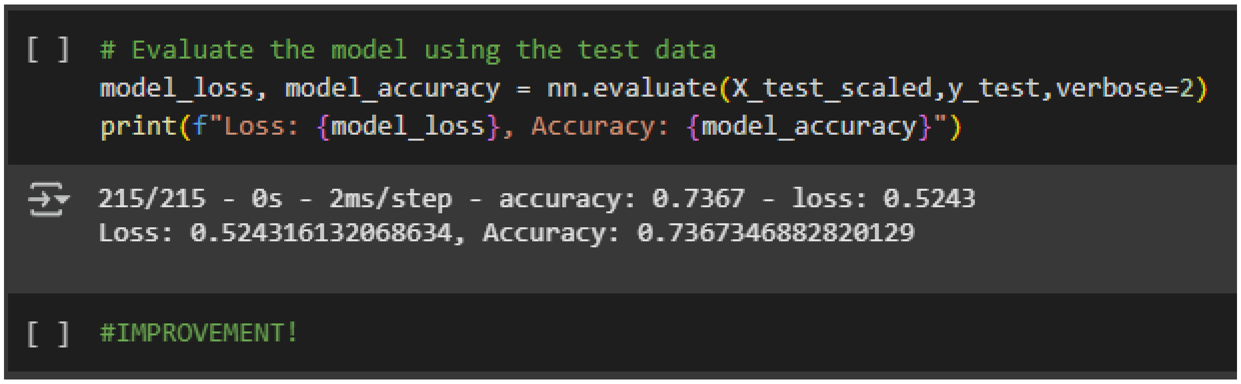
To achieve the target objective of 75%, optimization work first focused on adding an additional hidden layer and modifying the number of nodes in each layer to increase training parameters. Additionally, this round of modifications increased the number of epochs to facilitate deeper model analysis. While this approach sought to improve overall model accuracy, the output showed a decline to 72.62%, which indicated this modification approach pushed the model in a direction opposite of the target objective. Therefore, further optimization efforts deployed different strategies to push the model toward higher accuracy.

Secondary Optimization

The first step in ensuing optimization work included reduction of the nodes for each layer in the model and the inclusion of a 30% dropout function to reduce overfitting; however, these modifications did not improve model performance. As a result, optimization efforts removed these modifications. Following, a review of the dataset and the potential issues with the model showed a potential overfitting problem as the likely cause for the model's performance issues. This led to a theory that the relatively small size of the data could be inhibiting the model's performance, essentially creating a "mismatch" between data size and model complexity.

Tertiary Optimization

In turn, the third optimization attempt sought to expand the data included in the data frame by adding the "Name" column back into the data set and processing it via One Hot Encoding and Standard Scaling. Additionally, this optimization attempt sought to decrease noise by binning more aggressively on each feature. Further modifications sought to minimize overfitting risk by substantially reducing the number of nodes in each layer. This iterative attempt resulted in improved model performance, with output showing accuracy of 73.67%. Given this improvement, further optimization expanded on this method.



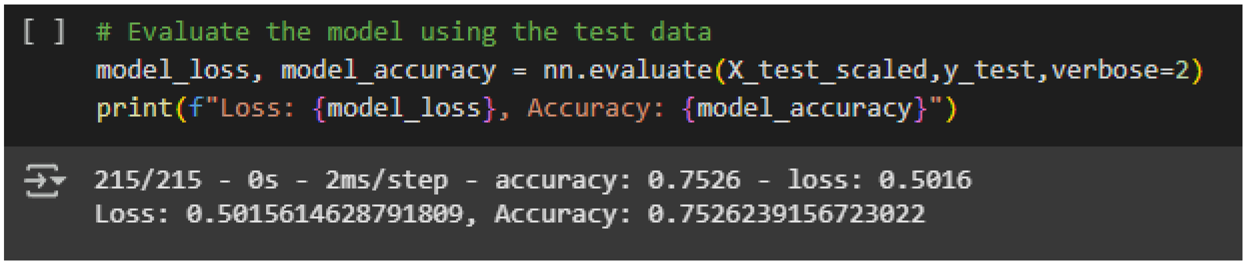
```
[ ]  # Evaluate the model using the test data
     model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
     print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
215/215 - 0s - 2ms/step - accuracy: 0.7367 - loss: 0.5243
Loss: 0.524316132068634, Accuracy: 0.7367346882820129
```

```
[ ]  #IMPROVEMENT!
```

Quaternary Optimization

In the final optimization attempt, modifications included more aggressive binning in each of the three features, a slight decrease in nodes for each layer, and heavier reliance on Sigmoid to further improve model accuracy. Additionally, the inclusion of the "Name" data remained in this attempt. Fortunately, the hypothesis developed in the previous optimization attempt proved valid following these changes. Model output accuracy improved to 75.26% after these modifications, producing a model that met the initial objectives of this analysis.



```
# 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}")
```

```
215/215 - 0s - 2ms/step - accuracy: 0.7526 - loss: 0.5016
Loss: 0.5015614628791809, Accuracy: 0.7526239156723022
```

Summary:

Overall, the optimized model met the objectives in this instance; however, the model still has ample room for improving accuracy. Further modification to the layer nodes and greater consistency in the use of model functions may help meet that objective. That stated, the model's current performance will improve assessment of applicant future financial performance at Alphabet Soup, which will assist the organization in its selection of funding recipients. As a result, each funding dollar will have a greater chance of being allocated to organizations that will prosper and generate positive impact in the community and local economy.