Sally Ahmed
5/5/2022

Deep Learning Project: Charity Funding Prediction

The non-profit foundation Alphabet Soup wants to create an algorithm to predict whether or not applicants for funding will be successful. With our knowledge of machine learning and neural networks, we'll use the features in the provided dataset to create a binary classifier that is capable of predicting whether applicants will be successful if funded by Alphabet Soup.

From Alphabet Soup's business team, We have received a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. Within this dataset are a number of columns that capture metadata about each organization, such as the following:

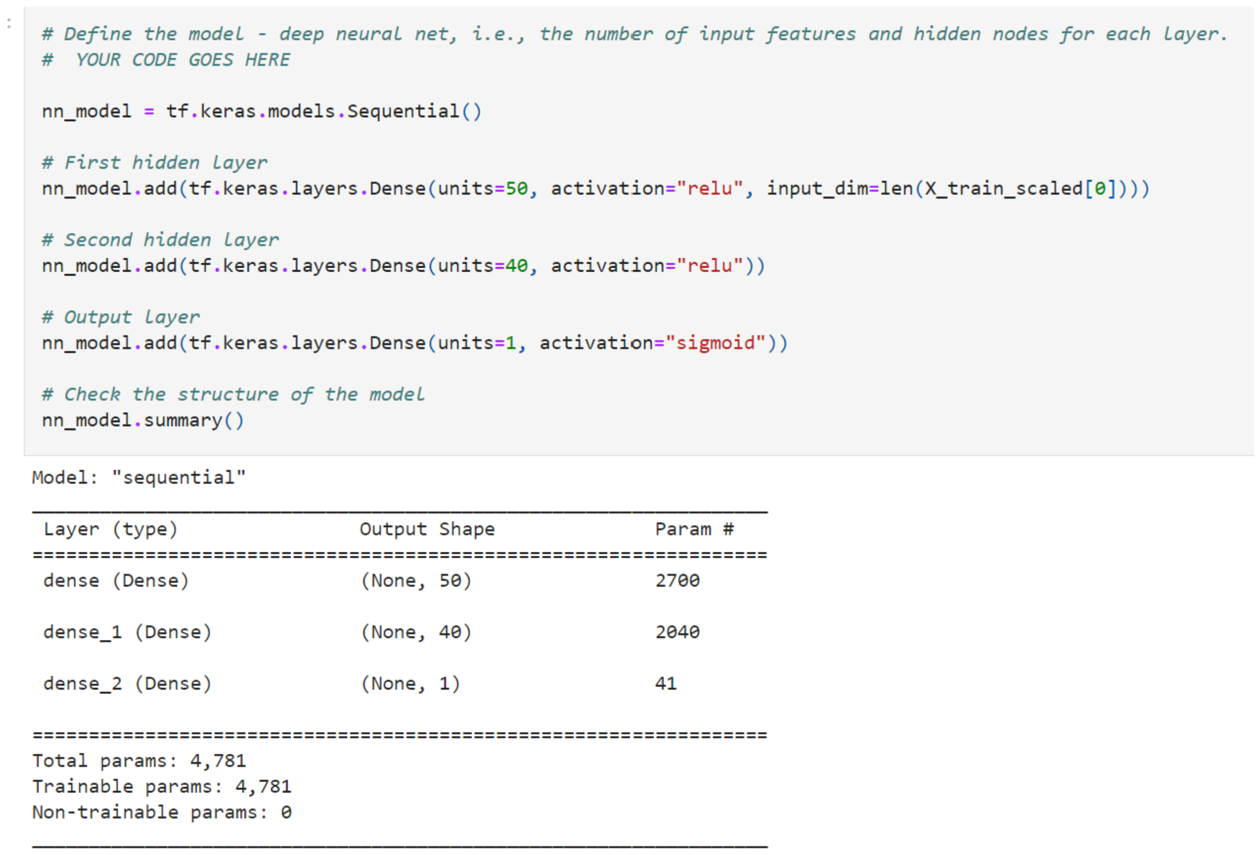
Data Processing:

The dataset removed any irrelevant information's there fore EIN and Name were dropped from the model. The remaining columns were considered features for the model. Although NAME was added back in the second test. CLASSIFICATIONS and APPLICATION_TYPE was replaced with "Other" due to high fluctuation. The data was split into training and testing sets of data. The target variable for the model is "IS_SUCCESSFUL" and is verified by the value, 1 was considered yes and 0 was no. APPLICATION data was analyzed, and CLASSIFICATION's value was used for binning. Each unique value used several data point was 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(). We scaled the optimized data.

Compiling, Training and Evaluation the Model

Neural Network was applied on each model multiple layers, three in total. The number of features dictated the number of hidden.

Sally Ahmed
5/5/2022



```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
#  YOUR CODE GOES HERE

nn_model = tf.keras.models.Sequential()

# First hidden layer
nn_model.add(tf.keras.layers.Dense(units=50, activation="relu", input_dim=len(X_train_scaled[0])))

# Second hidden layer
nn_model.add(tf.keras.layers.Dense(units=40, activation="relu"))

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

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

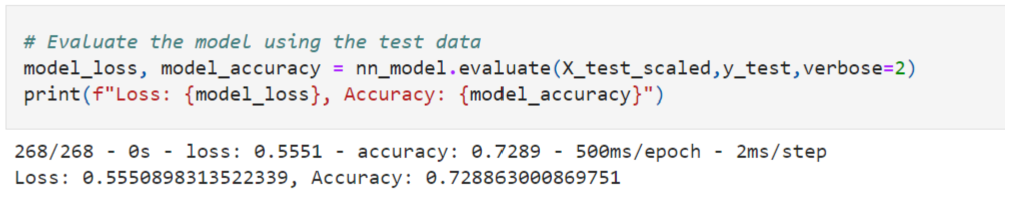
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                2700

 dense_1 (Dense)             (None, 40)                2040

 dense_2 (Dense)             (None, 1)                 41

=================================================================
Total params: 4,781
Trainable params: 4,781
Non-trainable params: 0
_________________________________________________________________
```

A three_layers training model generated 2700 parameters. The first attempt came close to 0.72% which was under the desired 75%



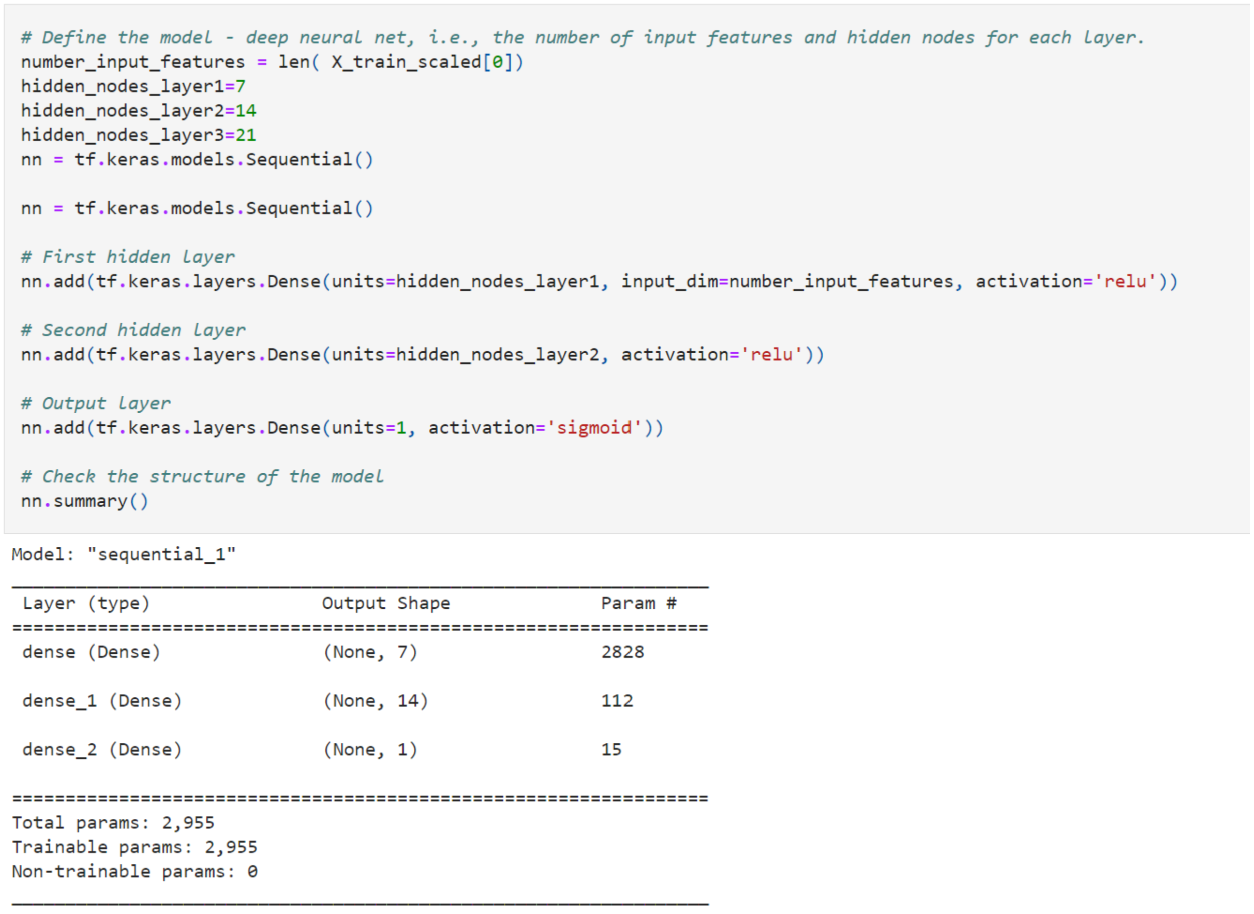
```
# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.5551 - accuracy: 0.7289 - 500ms/epoch - 2ms/step
Loss: 0.5550898313522339, Accuracy: 0.728863000869751
```

Optimization:

Summarize the overall results of the deep learning model. The Second attempt added "Name" back into the dataset, we scaled the data. this time I achieved .78% which was 3% over target. A total of 2955 Params.

Sally Ahmed
5/5/2022



```
# 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()

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()
```

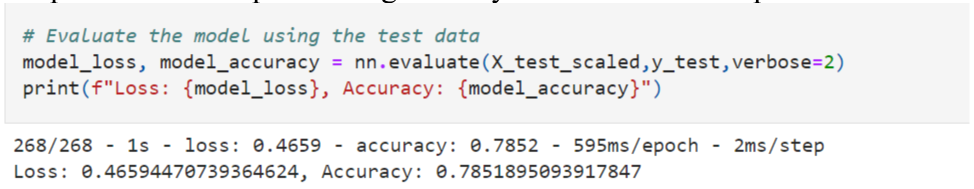
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 2828

 dense_1 (Dense)             (None, 14)                112

 dense_2 (Dense)             (None, 1)                 15

=================================================================
Total params: 2,955
Trainable params: 2,955
Non-trainable params: 0
_________________________________________________________________
```

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.



```
# 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: 0.4659 - accuracy: 0.7852 - 595ms/epoch - 2ms/step
Loss: 0.46594470739364624, Accuracy: 0.7851895093917847
```