Deep Learning Project: Charity Funding Predictor

Author: Melina Heredia

Overview

This project consisted of creating a neural net machine learning model to predict the success rate of a venture paid by the non-profit foundation Alphabet Soup. Utilizing the features provided in the CSV dataset, which contains more than 34,000 organizations that have received funding from Alphabet Soup over the years, a binary classifier is created that in essence will be capable of predicting whether Alphabet Soup foundation applicants will be successful.

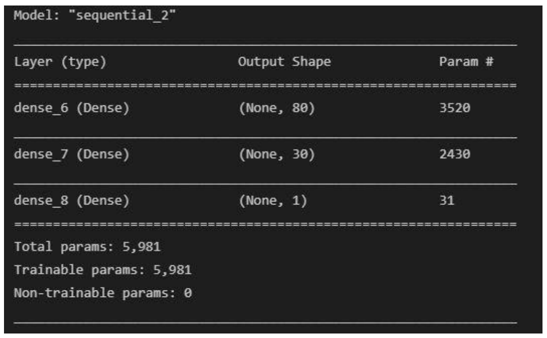
Results

Data Preprocessing:

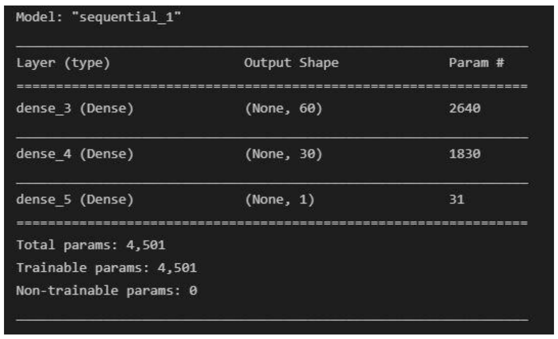
- The variable "IS_SUCCESSFUL" is the target for this model.
- The feature variables for this model originally included all but the "EIN" and "NAME" variables.
- Both the "EIN" and "NAME" variables were neither target nor feature variables and were so both removed from the input data.

Compiling, Training, and Evaluating the Model

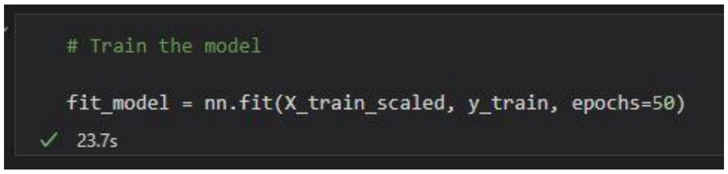
- Two hidden layers and two activation functions were selected when attempting to optimize the model. 43 input features were used and a combination 60/30 and 80/30 neurons were used when attempting to optimize.
- A multilayer network was chosen to overcome the limitation of linear separability, and two activation functions were ultimately used during experimentation when attempting to achieve target model performance.
- Target model performance was not reached, despite the many attempts and experimentation.
- A facet of steps were taken to increase model performance of which included decreasing and increasing the number of nodes, increasing and decreasing the number of epochs, and utilizing different combinations of activation functions.



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 80)                3520
_________________________________________________________________
dense_7 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 31
=================================================================
Total params: 5,981
Trainable params: 5,981
Non-trainable params: 0
_________________________________________________________________
```

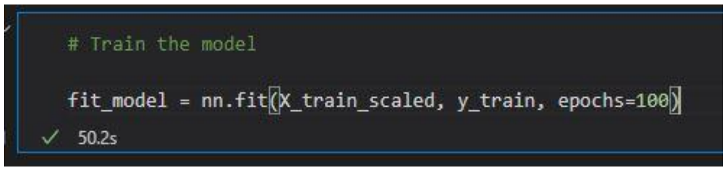


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 60)                2640
_________________________________________________________________
dense_4 (Dense)              (None, 30)                1830
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 31
=================================================================
Total params: 4,501
Trainable params: 4,501
Non-trainable params: 0
_________________________________________________________________
```



```
# Train the model

fit_model = nn.fit(X_train_scaled, y_train, epochs=50)
✓ 23.7s
```

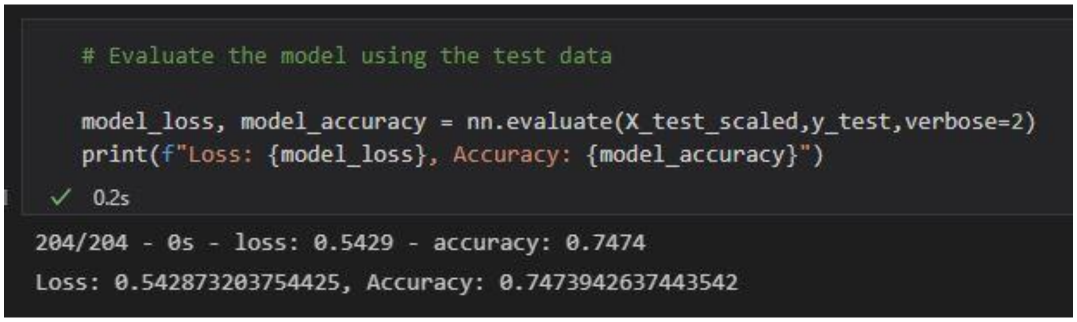


```
# Train the model

fit_model = nn.fit(X_train_scaled, y_train, epochs=100)
✓ 50.2s
```

Summary

Overall, while target performance was not reached, the most accurate model had a success rate of 0.7474, just a hair off the desired 0.75. The inability of this model to reach desired results could be a result of a lack of data. Neural net models typically require larger datasets to achieve higher accuracy success rates. Thus, to improve the goal of this project, one could instead use a simpler model such as a random forest classifier model. Such a model could result in higher accuracy, as the random forest classifier model relies on categorical features and a smaller dataset.



```
# 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}")
✓ 0.2s
204/204 - 0s - loss: 0.5429 - accuracy: 0.7474
Loss: 0.542873203754425, Accuracy: 0.7473942637443542
```