Lauren To
11/12/21

CharityFund Model Optimization

Overview

The goal of this project is to evaluate the effectiveness of neural networks and deep learning to predict whether applicants will be successful if funded by nonprofit, Alphabet Soup, based on data from over 34,000 organizations.

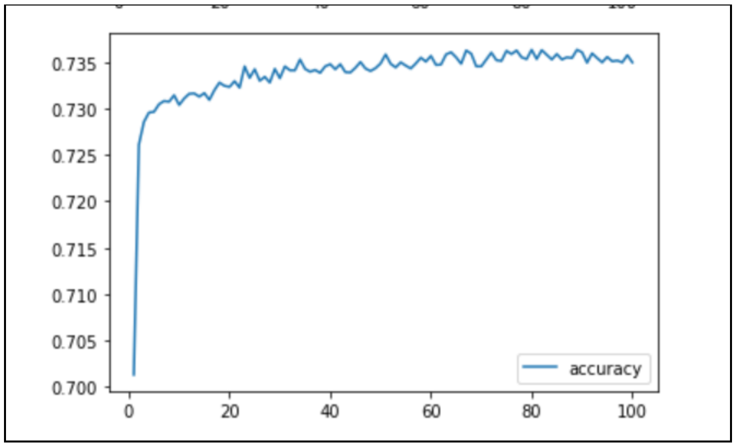
Modeling

- Columns EIN and NAME are used for identification, and do not contain relevant information for the model. These columns were dropped.
- I believe the rest of the columns are useful features to help predict applicant success. These columns were one-hot encoded, converted from categorical data to numeric with `pd.get_dummies.
- Due to high variance, values for CLASSIFICATION were replaced with 'Other' if less than 528 using binning. The same was done for APPLICATION_TYPE values less than 900.
- Resulting data included 44 features. The target variable (y) = IS_SUCCESSFUL.
- Data was split into training and test subsets.

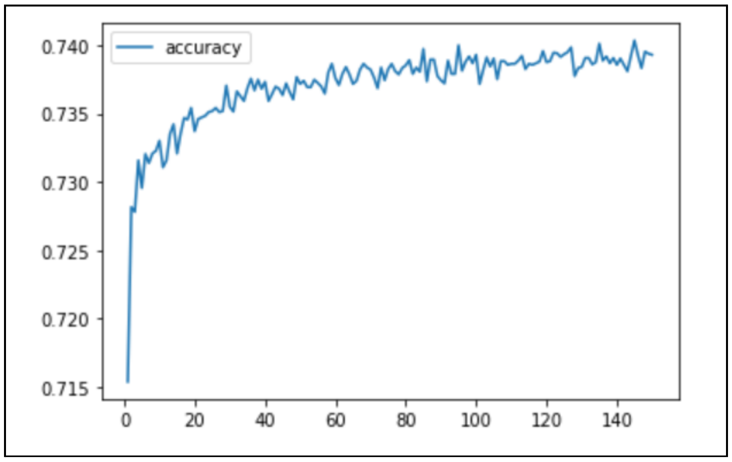
Results

The model needed to achieve a target predictive accuracy higher than 75%. I made three attempts using a neural network model, and they all fell slightly under. Results and hyperparameters from each attempt are detailed below:

- **The first model** (nn_charity_model.ipynb) resulted in an **accuracy score of 73.0%**. This means that 73% of the model's predicted values align with the dataset's true values. Hyperparameters were set arbitrarily to establish a baseline. The sigmoid function was used for the output layer because it outputs between (0,1) for our binary outcome. Hyperparameters used were:
    - *Layers* = 3
    - *Units* = 10 (layer 1), 15 (layer 2), 1 (output)
    - *Epochs* = 100
    - *Activation Function* = Relu

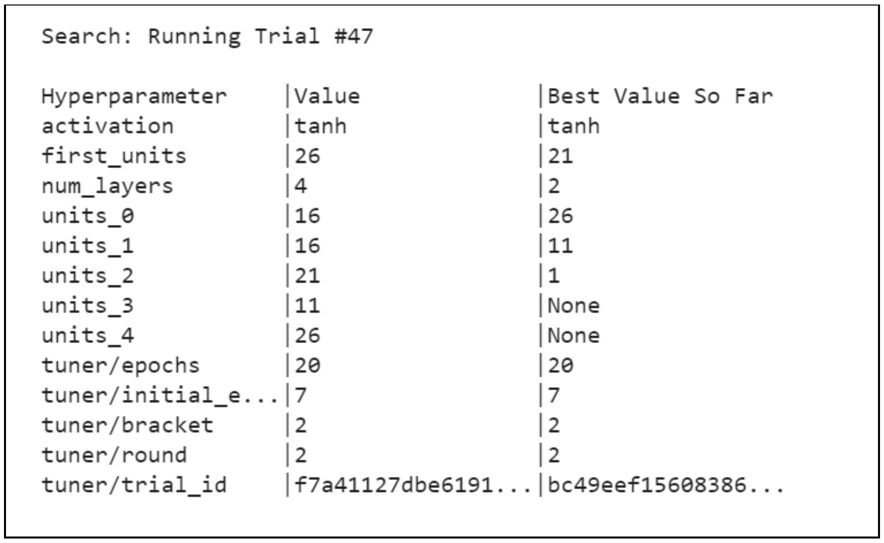


- **The second model** (nn_charity_model_v1.ipynb) resulted in an **accuracy score of 72.7%**. This means that 72% of the model's predicted values align with the dataset's true values. I tuned the following hyperparameters: increasing the number of layers, the number of neurons, and the number of epochs. I also added a 'tanh' activation function for layer 3, using relu for the rest of the layers:
  - *Layers* = 4
  - *Units* = 20 (layer 1), 18 (layer 2), 10 (layer 3), 1 (output)
  - *Epochs* = 150
  - *Activation Function* = Relu & Tanh (layer 3)



- **The third model** (nn_charity_model_v2.ipynb) was automated using **keras-tuner** to see if an accuracy score of 75% was possible. Keras-tuner helps to pick the optimal set of hyperparameters for a model using TensorFlow. I included a 'stop early' function where after 5 epochs with no improvement, training would be stopped. The resulting model from keras-tuner had a score of **73.3% accuracy**. This means that 73% of the model's predicted values align with the dataset's true values, which was slightly higher than the first model. Keras-tuner tuned hyperparameters as follows:
  - *Layers* = 2

- *Units* - 21 (layer 1), 1 (output)
- *Epochs* = 20
- *Activation Function* = Tanh (The tanh function allows for values that approach −1, outputs between (-1,1))



```
Search: Running Trial #47

Hyperparameter    |Value             |Best Value So Far
activation        |tanh              |tanh
first_units       |26                |21
num_layers        |4                 |2
units_0           |16                |26
units_1           |16                |11
units_2           |21                |1
units_3           |11                |None
units_4           |26                |None
tuner/epochs      |20                |20
tuner/initial_e...|7                 |7
tuner/bracket     |2                 |2
tuner/round       |2                 |2
tuner/trial_id    |f7a41127dbe6191...|bc49eef15608386...
```

Summary

In the three attempts I made, manually and using keras-tuner, **the model was unable to achieve a target predictive accuracy higher than 73.3%**. Hypertuning resulted in marginal improvements. However, given the binary nature of the problem, I would consider using another classification model such as Random Forest or Logistic Regression to predict whether applicants will be successful if funded by Alphabet Soup.

Learnings

With more time, I would:

- **Reduce the number of features** to help improve the model's accuracy score. Explore dropping features, or using Label Encoding over One-Hot Encoding to reduce the featureset.
- **Check for multicollinearity** using the Variance Inflation Factor (VIF) to see if dummy variables have VIF above 5, and there is a multicollinear problem.
- **Evaluate the model on other common metrics** for a binary classification problem such as precision, recall and f1-score.
- **Compare the neural network model with other classification models** like Logistic Regression and Random Forest to see if the accuracy score improves. A neural network may not be the best algorithm to solve this problem.