**Neural Network – Analysis and Optimization**

**Overview**

Alphabet Soup is a nonprofit organization that has funded a multitude of companies and business ventures. Over the years, they have amassed a significant amount of data about the companies they have become affiliated with. This data includes organization name, application type, use case, status, income classification, ask amount, and more. With this information, Alphabet Soup wants a tool that could successfully predict an applicant’s success if funding was provided. An applicant is considered successful if the provided money is used in an effective manner. To fulfill this request, a neural network was established and optimized to an acceptable level of accuracy.

**Results**

Data Preprocessing

The target for the model is the “IS\_SUCCESSFUL” column, which ultimately judges if the granted money was used in an effective manner.

The features and their descriptions are the following:

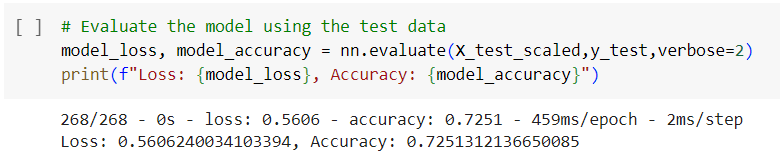
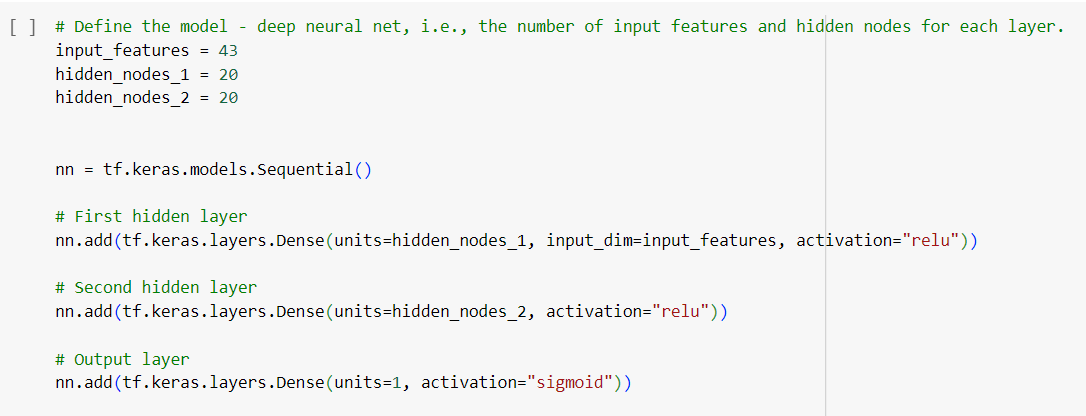
* APPLICATION\_TYPE - Alphabet Soup application type
* AFFILIATION - Affiliated sector of industry
* CLASSIFICATION - Government organization classification
* USE\_CASE - Use case for funding
* ORGANIZATION - Organization type
* STATUS - Active status
* INCOME\_AMT - Income classification
* SPECIAL\_CONSIDERATIONS - Special considerations for application
* ASK\_AMT - Funding amount requested

For the original model, the EIN and NAME columns were dropped because they are neither features nor columns. However, for the optimized model, maximum accuracy was achieved when the NAME column was included, and the following were dropped: EIN, SPECIAL\_CONSIDERATIONS, USE\_CASE, STATUS.

Compiling, Training, and Evaluating the Model

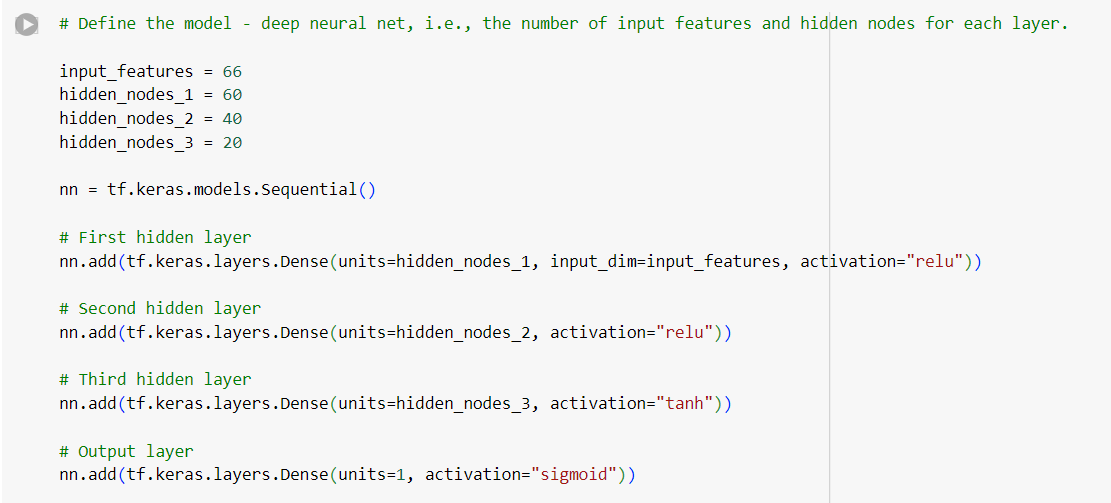
The original model only scored 72.5% in accuracy and was unable to reach the target performance of 75%. The specifications for the model are:

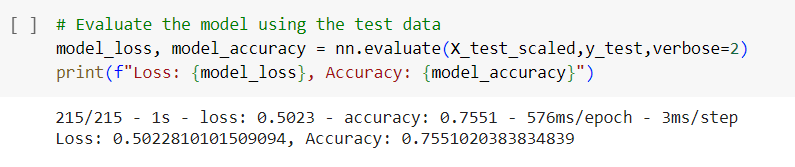
* 43 input features
* 2 hidden layers
* Relu activation function for both layers
* 20 neurons for each hidden layer
* Sigmoid activation function for the output layer
* 50 epochs
* Loss of 56%



The optimized model fared better, achieving 75.5% accuracy. Its specifications are:

* 66 input features
* 3 hidden layers
* Relu activation for first two hidden layers
* Tanh activation for final hidden layer
* 60, 40, and 20 neurons for each respective hidden layer
* Sigmoid activation for the output layer
* 100 epochs
* Loss of 50.2%





As previously mentioned, the NAME column was included in this version of the model while EIN, SPECIAL\_CONSIDERATIONS, USE\_CASE, STATUS was dropped. This seemed to help the model’s accuracy significantly. Including the name column, however, may have led to some overfitting, which may have contributed to the increase in accuracy. Other factors that contributed to the increase in accuracy could be that the training data set was increased to 80% of the total available data, a hidden layer was added, different activation functions were used in these layers, the number of neurons per layer were increased, and the number of epochs were doubled.

**Summary**

In conclusion, the unoptimized neural network was able to accurately predict approximately 72% of the test data. The optimized model’s improved accuracy (75.5%) could be due to a variety of factors, including an extra layer, more neurons, use of various activation factors, increased epochs, readdition of the “NAME” column, exclusion of other columns, and a slightly greater amount of training data.

Another machine learning model that would work well with this particular scenario would be a random forest algorithm. Random forests are effective at minimizing overfitting and outliers. In addition, they are easily interpretable and are efficient at handling classification tasks such as this one. Perhaps most importantly, it requires far less computational power than neural networks, making it ideal if resources are limited.