**Report on the Neural Network Model**

The purpose of this analysis is to create an algorithm to predict whether or not applicants for funding will be successful. Machine Learning and neural networks will be used to create a binary classifier that is capable of predicting whether applicants will be successful if funded.

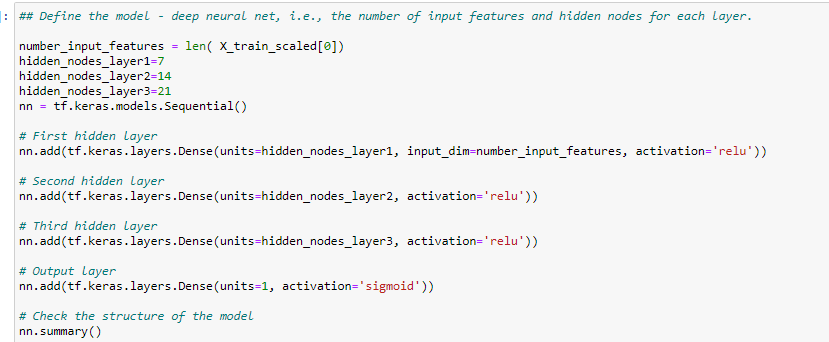
Data Preprocessing

The target variable for the model will be "IS\_SUCCESSFUL". This column has values of 1 and 0 which helps us determine if the charity fund is successful (1) or not successful (0). After columns "EIN" and "Name" are dropped, the remaining columns are features for the model.

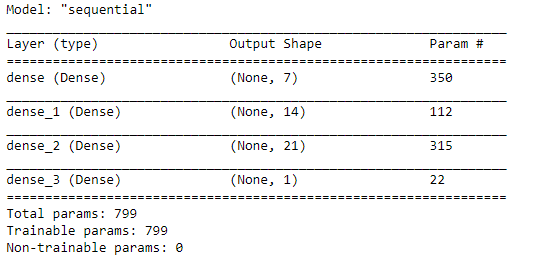
Dropped the EIN and NAME columns then determined the number of unique values for each column. Looked at APPLICATION value counts, and CLASSIFICATION Value counts for binning. Used the number of data points for each unique value to pick a cutoff point to bin "rare" categorical variables together in a new value, `Other`, and then check if the binning was successful. Use `pd.get\_dummies()` to encode categorical variables.

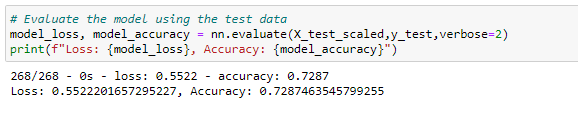
**Compiling, Training, and Evaluating the Model**

Created neural network with three hidden layers as shown below. The number of hidden nodes were selected based on the number of features initially. The number of input features and hidden nodes for each feature shown below.



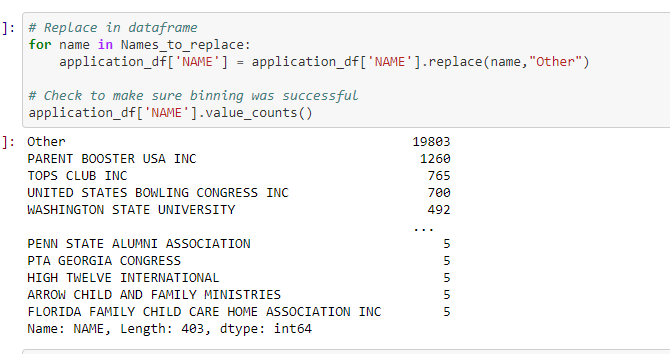
Generated total of 799 parameters to train model. I was able to get an accuracy of 0.7285 for the first model.

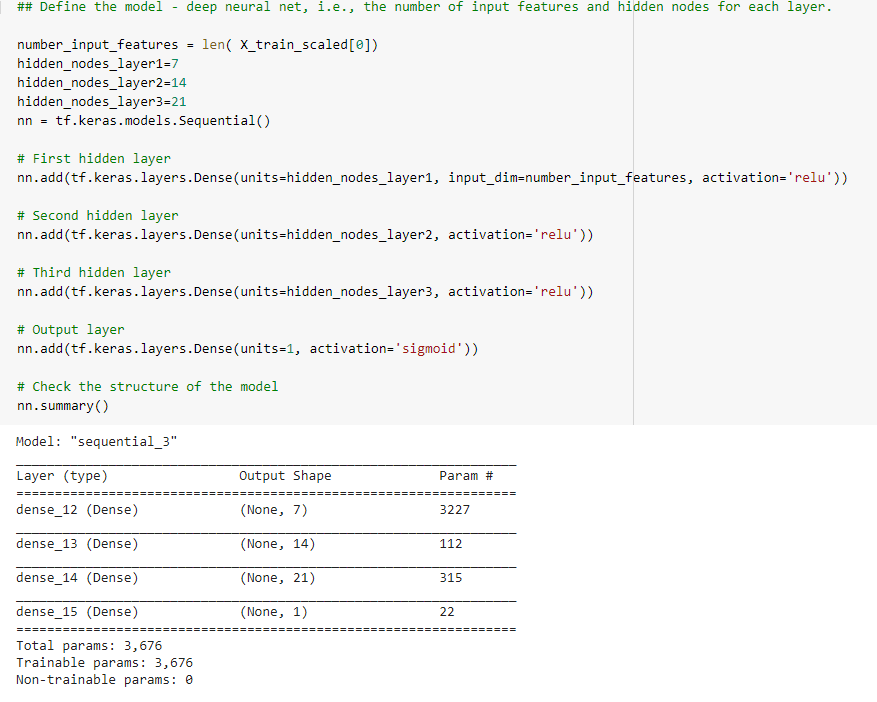




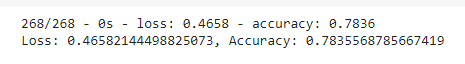
Optimization

Using the original Jupyter notebook I decided to put NAME column back in the model. I looked at NAME value type for binning. The count of the NAME indicated how many times an organization was funded. For all NAME less than 5 added to Other category. Name Column went from Length: 19568 to Length: 403.



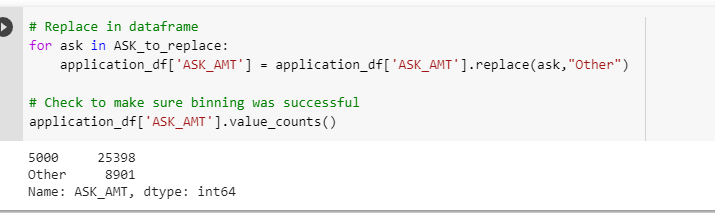
My reasoning for including the NAME back in model was I thought that having received funding before would be a predictor of future funding. 

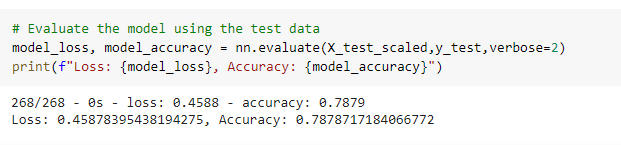
**Parameters went for 799 in first model to 3,676 and accuracy improve from 0.728 to 0.783 respectively.**

****

**Second Optimization;**

I decided to bin based on ASK\_AMT and followed the same steps as above. The accuracy improved to 0.7878.





**Summary**: Overall I was able to get an accuracy of 0.7878 using trial and error. This involved binning the following columns:

* APPLICATION\_TYPE
* CLASSIFICATION
* NAME
* ASK\_AMT

In summary, it can be concluded that it is recommended for a deep learning model to have multiple layers and large enough number of neuron, and the right activation function to reach the accuracy for a binary classification models.