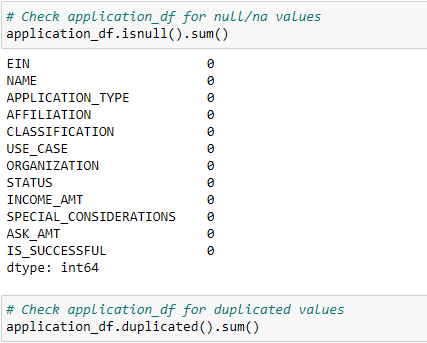
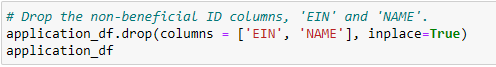
**1: Data Preprocessing**

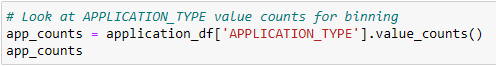
* Dataset was checked for null and duplicated values

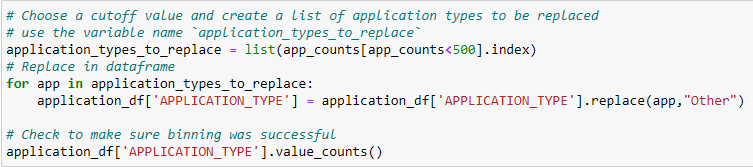


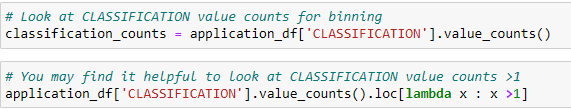
* **EIN** and **NAME**—Identification columns removed from the input data because they are neither targets nor features

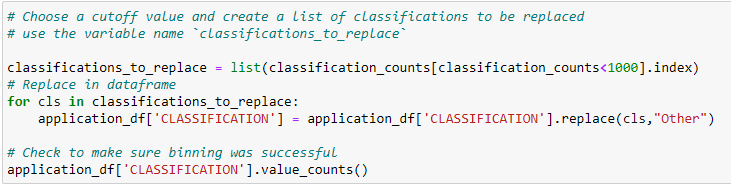


* Created cutoff point to bin "rare" categorical variables together in a new value, Other for both CLASSIFICATION and APPLICATION\_TYPE

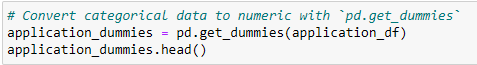


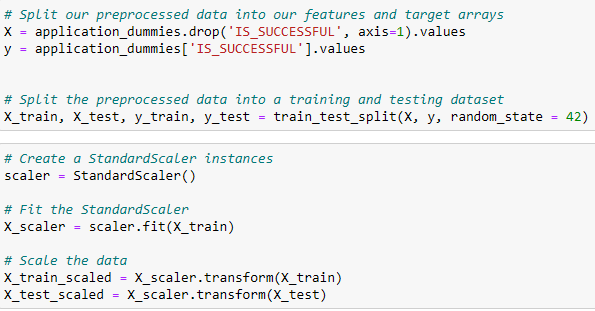






* Converted categorical data to numeric with pd.get\_dummies, split the preprocessed data into features and target arrays, then lastly split into training and tesing datasets





Target Variable for the model:

* IS\_SUCCESSFUL

Feature Variables for the model:

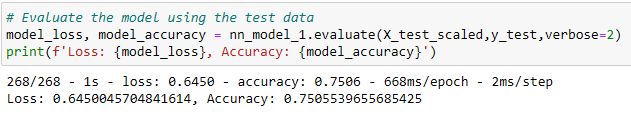
* APPLICATION\_TYPE
* AFFILIATION
* CLASSIFICATION
* USE\_CASE
* ORGANIZATION
* STATUS
* INCOME\_AMT
* SPECIAL\_CONSIDERATIONS
* ASK\_AMT

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

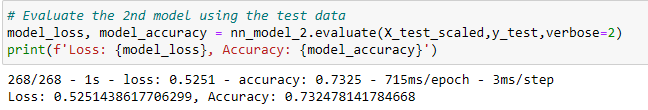
I build the first model with the following parameters with low computation time in mind:

* 2 hidden layers with 80, 30 neurons split (the input (node) feature was 43, 80 was chosen as the first layer as it is almost double the input feature). With an hidden layer activation function of rely as this our go to for first model.
* Output node is 1 as it was binary classifier model with only one output: was the funding application successful yes or no. And an output layer activation of sigmoid as the model output is binary classification between 0 and 1.

I then increased the hidden layers to 3 and set the third hidden layer at 30 as the model prediction accuracy was below 75%:



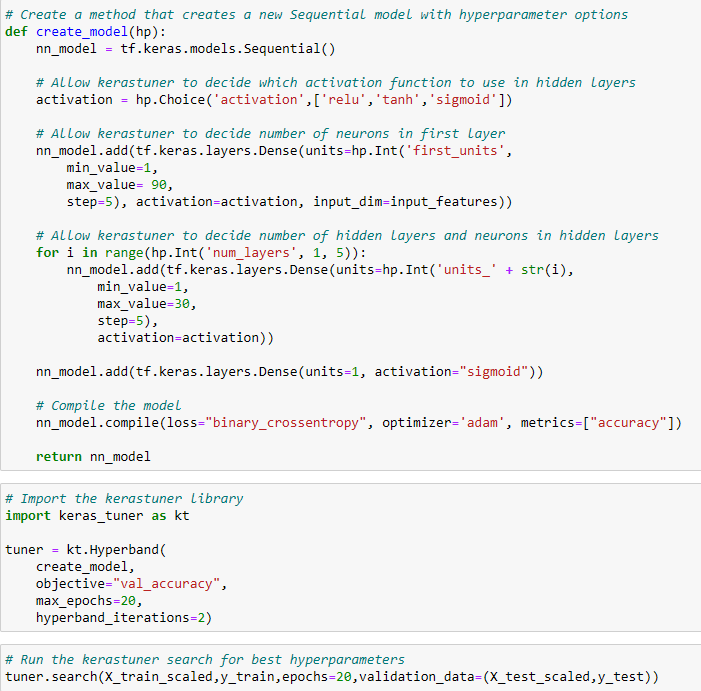
For the second model I decided to use tanh activation and 3 hidden layers with 90, 30, 20 neurons split and a sigmoid activation for output as the output doesn't change.



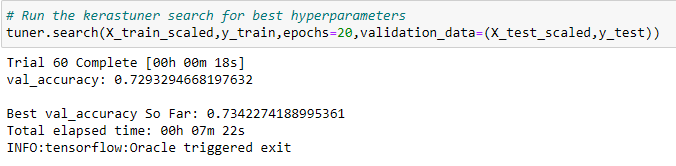
I experimented with increasing nodes and neurons, the accuracy of the first model is over 75%.

### 3: Optimize the Model

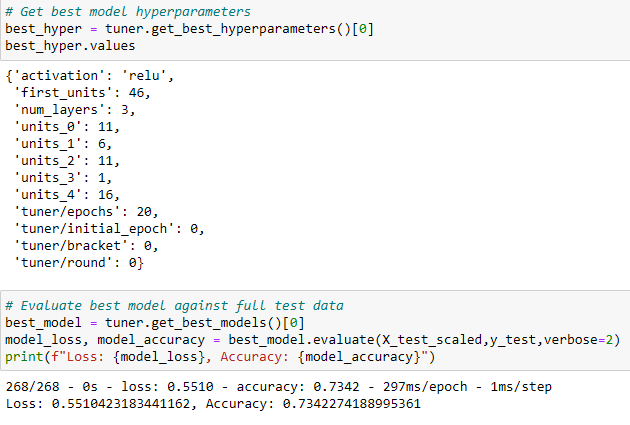
I decided to use an automated model optimizer to get the most accurate model possible by creating method that creates a keras Sequential model using the keras-tuner library with hyper-parameters options.



Which will automatically tune the hyper-pyrometers until it gets the most accurate model.

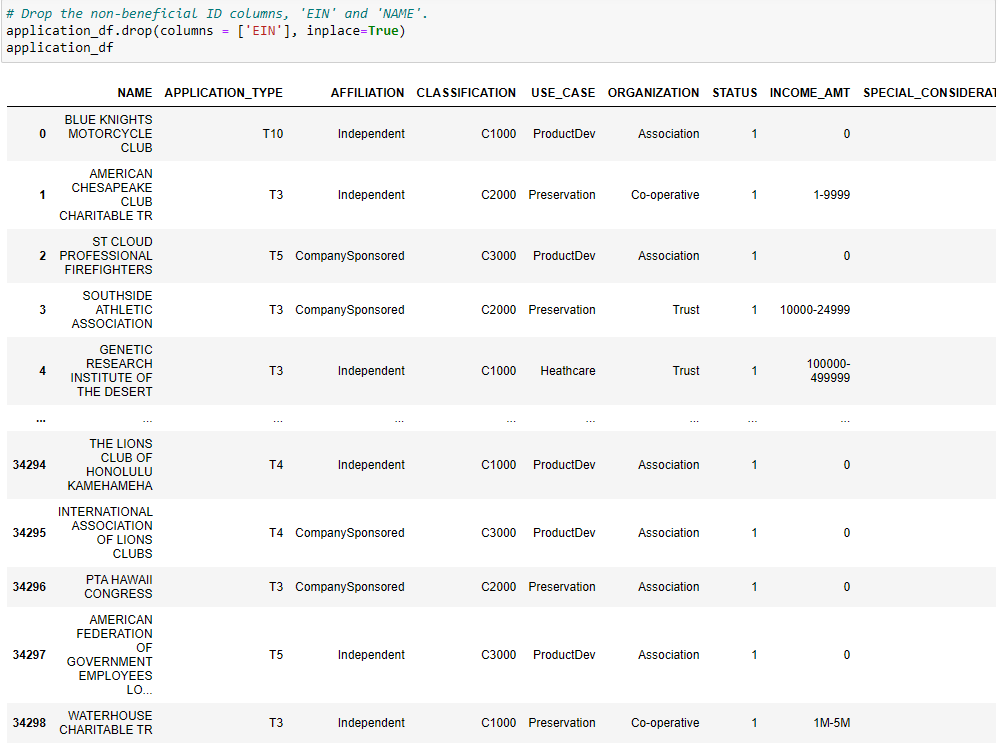


The best model from the keras tuner method achieved 73% prediction accuracy using a sigmoid activation function with input node of 46, 6 hidden layers at a 51, 81, 71, 6, 41, 91 neurons split and 100 training epochs.

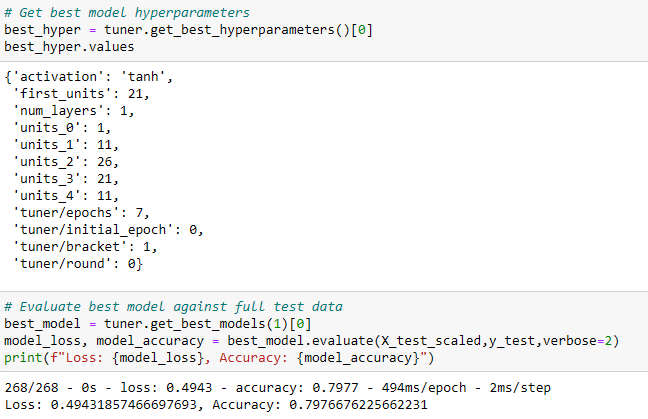


# **4. Final Optimization**

I kept the Name column for my final Optimized Model as I still hadn't reached the goal of 75% accuracy. Keping the keras-tuner the same apart from lowering the epochs from 100 to 50 for time optimization.

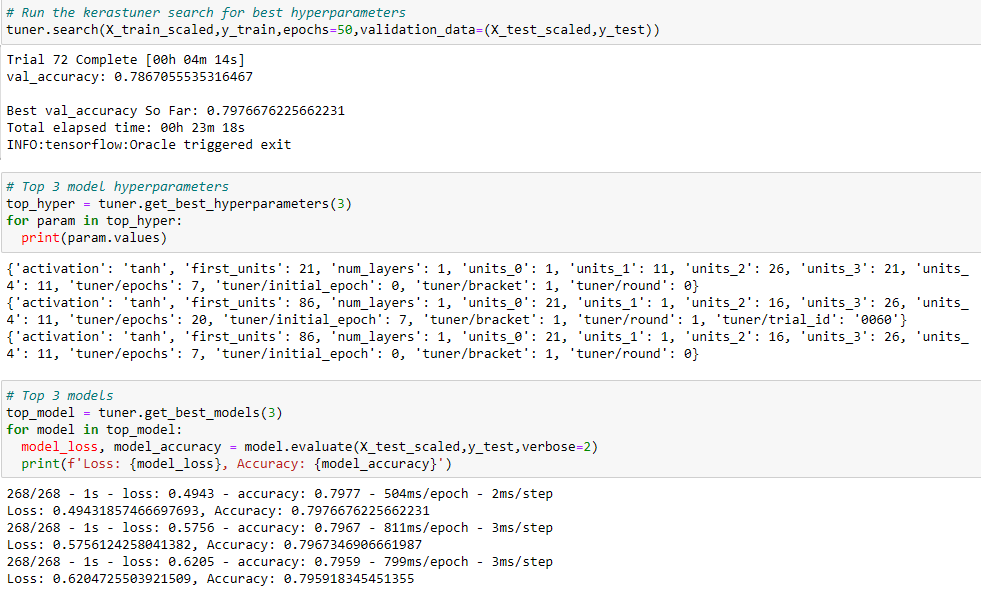


## Final and best optimized model achieved 80% accuracy , which exceeds the 75% goal and is our best model.



### All Top 3 Models were around the 80%.

All had a sigmoid activation with different number of input nodes and hidden layers:



# Summary:

The final automatically optimized neural network trained model from the keras tuner method achieved 80% prediction accuracy with a 0.45 loss, using a sigmoid activation function with input node of 76; 5 hidden layers at a 16, 21, 26, 11, 21, neurons split and 50 training epochs. Performing better than the non automized model. Keeping the Name column was crucial in achieving and going beyond the target. This shows the importance of the shape of your datasets before you preprocess it.