**Neural Network Model Report**

* **Overview**
  + This analysis aims to assist the nonprofit foundation Alphabet Soup in determining whether applicants for funding have a good chance at success in their ventures.
* **Results**:
  + Preprocessing
    - The target variable for this model would be the “IS\_SUCCESSFUL” column, as that column contains the information this model is trying to predict.
    - A screenshot of a black and white screen

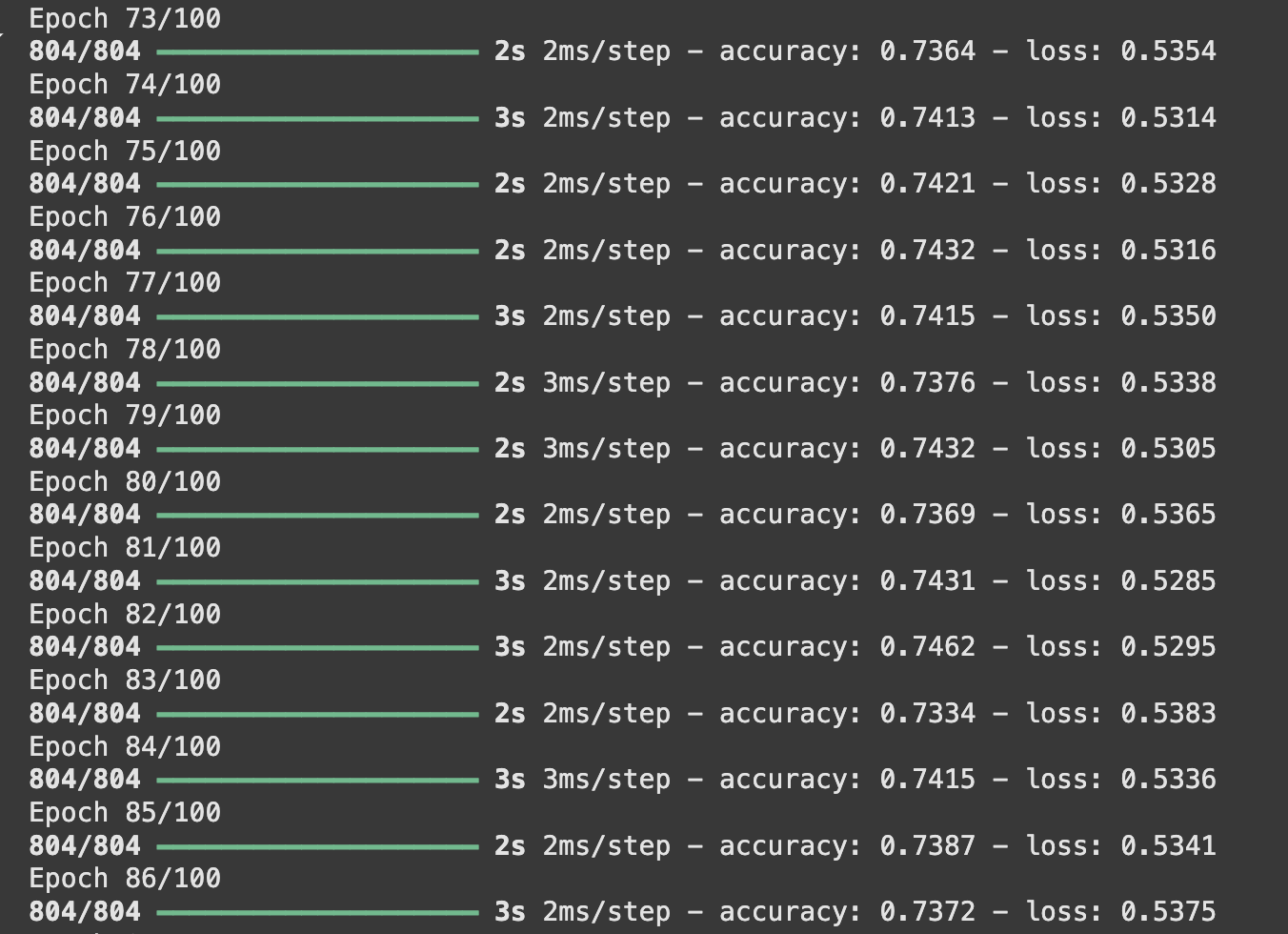
      Description automatically generatedThe variables that are considered features for this data would be the remaining columns in the data frame, excluding “IS\_SUCCESSFUL,” and therefore are the columns named “APPLICATION\_TYPE,” “AFFILIATION,” “CLASSIFICATION,” “USE\_CASE,” “ORGANIZATION,” “STATUS,” “INCOME\_AMT,” “SPECIAL\_CONSIDERATIONS,” and “ASK\_AMT.”
    - The variables that were removed from this data frame because they are not classified as either targets or features were “NAME” and “EIN,” as those values were not needed for the model to run.
  + Compiling, Training, and Evaluating the Model
    - For my sequential model, I utilized three hidden layers and one output layer. I decided to add a third hidden layer to try and improve the model’s accuracy while also being careful to avoid overfitting, which is why I kept the neurons relatively low on all of the layers. To avoid affecting the accuracy negatively, though, I decided to avoid changing up the activation functions and left them at their default functions.
      * First hidden layer: 40 neurons, rectified linear unit activation function
      * Second hidden layer: 20 neurons, rectified linear unit activation function
      * Third hidden layer: 10 neurons, rectified linear unit activation function
      * Output layer: 1 neuron, sigmoid activation function

A screen shot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

* + - Unfortunately, after around five different attempts at increasing the model’s performance, I was unable to achieve a 75% accuracy. After the first two or three attempts at doing so, it ended up plateauing at around 72%, and to avoid making the model’s accuracy worse, I made the decision to leave it at 72% as I noticed the more changes I made, the lower the number went.



* + - To try and increase the model’s performance, I attempted a few things, not necessarily in the order they were attempted. I increased the units for my first hidden layer up to 120, but that only brought the accuracy up one point, so I moved that back down to 40, where it originally was. I also increased the units in the second layer to 50, but that did not affect the accuracy. I also added a third hidden layer, but that brought the accuracy down. I left it, though, as I was struggling to increase the accuracy using other methods. I also both increased and decreased the number of epochs as well, but I was concerned about overfitting, so I brought the number of epochs back down to 100. One of the last methods I tried was utilizing validation split, which split the training data and left around 20% of it to evaluate the model’s performance while training. That, unfortunately, did not have much of an effect on the data either. In the end, I ended with an accuracy of 72.8% and a loss of 55%, which also isn’t ideal.
* **Summary**
  + Overall, my model unfortunately did not reach the ideal performance of 75%, and to try to prevent overfitting or underfitting, I made the decision to leave the model at the final accuracy of 72.8%. It is also important to note that there was still a large loss, at 55%. I believe it may be worth trying a simpler model to determine the classification problem, such as logistic regression or decision trees. Sometimes, in cases such as these, simpler is better.