I. Overview

This analysis is done for the non-profit organization 'AIphabet Soup' which wants to determine the applicants for funding with the best chance of success.
The idea is to create a tool that the organization can use to determine if a new funding applicant would be successful with the funding they receive.
To aid the creation of this tool, AIphabet Soup has provided a dataset containing records of 34,000 organizations that have received funding from them.

This tool needs to have a prediction rate of 75% or higher in order for it to be used.

II. Data Preprocessing

The column titled ‘IS_SUCCESSFUL’ is the target for the model.

The following list of columns outlines the features of the model:

1. STATUS
2. ASK_AMT
3. APPLICATION_TYPE_Other
4. APPLICATION_TYPE_T19
5. APPLICATION_TYPE_T3
6. APPLICATION_TYPE_T4
7. APPLICATION_TYPE_T5
8. APPLICATION_TYPE_T6
9. AFFILIATION_CompanySponsored
10. AFFILIATION_Family/Parent
11. AFFILIATION_Independent
12. AFFILIATION_National
13. AFFILIATION_Other
14. AFFILIATION_Regional
15. CLASSIFICATION_C1000
16. CLASSIFICATION_C1200
17. CLASSIFICATION_C2000
18. CLASSIFICATION_C2100
19. CLASSIFICATION_C3000
20. CLASSIFICATION_Other
21. USE_CASE_CommunityServ
22. USE_CASE_Heathcare
23. USE_CASE_Other
24. USE_CASE_Preservation
25. USE_CASE_ProductDev
26. ORGANIZATION_Association
27. ORGANIZATION_Co-operative

28. ORGANIZATION_Corporation
29. ORGANIZATION_Trust
30. INCOME_AMT_0
31. INCOME_AMT_1-9999
32. INCOME_AMT_10000-24999
33. INCOME_AMT_100000-499999
34. INCOME_AMT_10M-50M
35. INCOME_AMT_1M-5M
36. INCOME_AMT_25000-99999
37. INCOME_AMT_50M+
38. INCOME_AMT_5M-10M
39. SPECIAL_CONSIDERATIONS_N
40. SPECIAL_CONSIDERATIONS_Y

What variable(s) should be removed from the input data because they are neither targets nor features?

The following columns are to be removed from the dataset because they do not contain relevant information for the model: EIN, NAME.

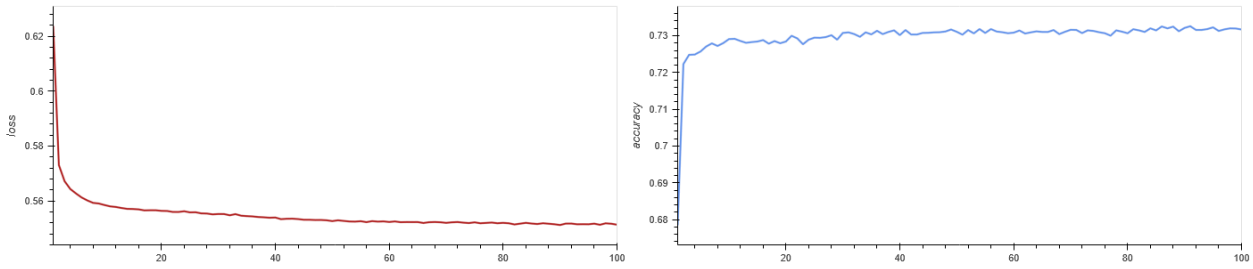
III. Compiling, Training, and Evaluating the Model

During the model creation, I did not end up finding a model that met the requirements of the organization (75% acc or higher). I will outline these attempts below.

- **First Model**

Layers of the first model are as follows:
1. 7 neurons with a 'relu' activation.
2. 1 neuron with a 'sigmoid' activation.

I chose a simple model because I believe that it would be able to solve the problem at hand and it would require the smallest amount of computing power and space. However, I was proven wrong.



Final results of the first model were a loss of 0.558 and an accuracy of 0.725.

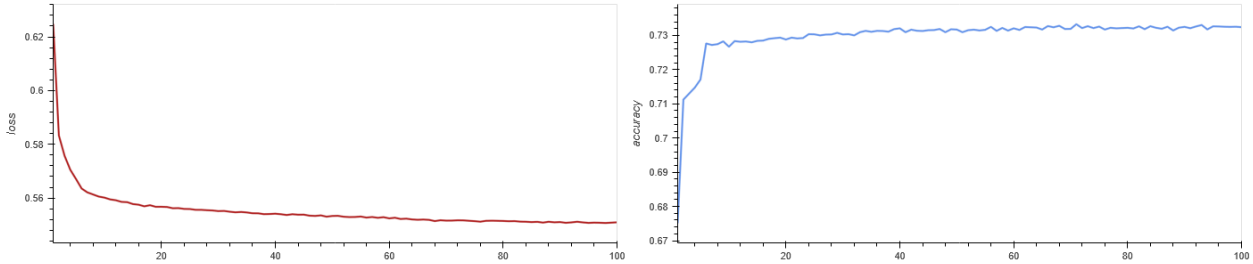
- **Second Model**

After the failure of the first model, I decided to use a more advanced model with one more layer and several more neurons.

Layers of the second model are as follows:

1. 5 neurons with a 'relu' activation.
2. 3 neurons with a 'relu' activation.
3. 1 neuron with a 'sigmoid' activation.

This model performed similarly to the first model.



Final results of the second model were a loss of 0.555 and an accuracy of 0.728.

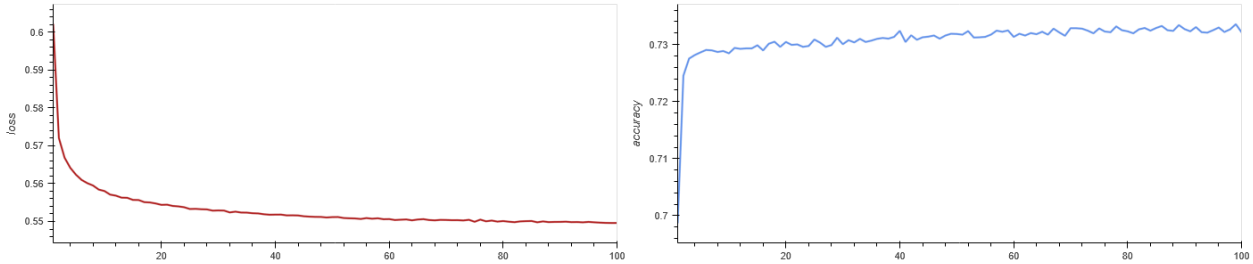
- **Third Model**

After another failure with the second model, I decided to change the activation function on the layers. I did not know what to expect from this change but I figured that it would be worth a shot changing it. I decided to use the tanh function for the activation layers.

Layers of the third model are as follows:

1. 5 neurons with a 'tanh' activation.
2. 3 neurons with a 'tanh' activation.
3. 1 neuron with a 'sigmoid' activation.

This model performed under the target as well but it did increase performance a little bit.



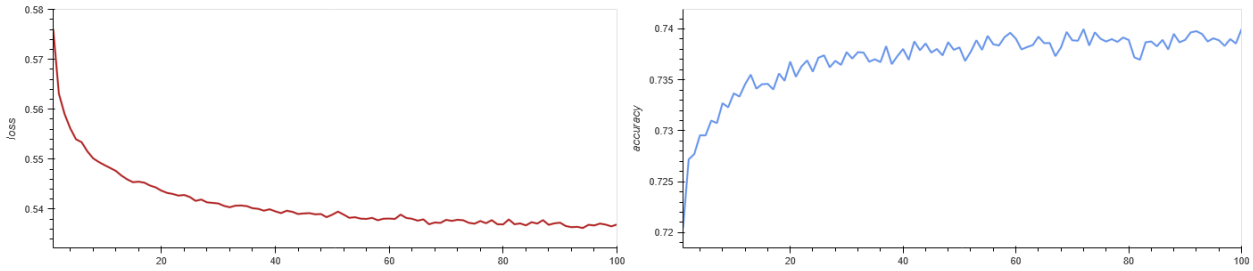
Final results for the third model were a loss of 0.559 and an accuracy of 0.730.

- **Fourth Model**

Given the minor success with the tanh activation function, I decided to keep that change. But there was still another change that had to be made in order to meet the goal. I decided to vastly increase the amount of neurons and add an additional layer.

Layers for the fourth model are as follows:
1. 100 neurons with a 'tanh' activation.
2. 100 neurons with a 'tanh' activation.
3. 100 neurons with a 'tanh' activation.
4. 1 neuron with a 'sigmoid' activation.

This model did not perform better than the third model, but it was quite close in performance so it seems



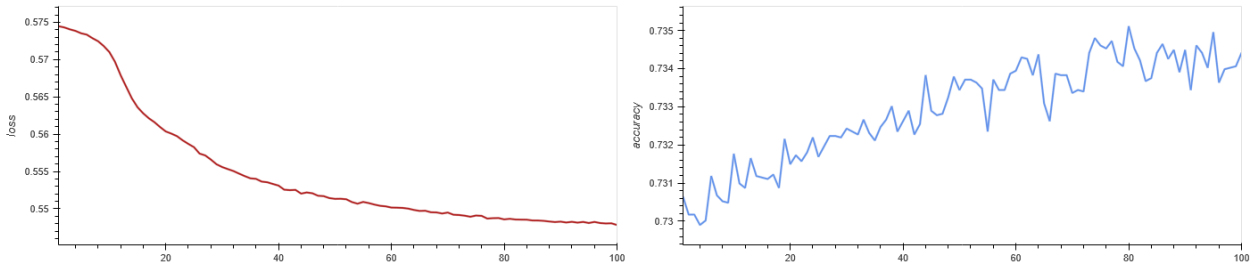
Final results for the fourth model were a loss of 0.565 and an accuracy of 0.729.

- **Fifth Model**

For the fifth model that I tried, I decided to use the keras tuner hyperparameter tuning tool. I did this because I figured it would find the optimal configuration for the model through its tuning.

Following are the parameters for the best model:
1. 'activation': 'sigmoid',
2. 'first_units': 7,
3. 'num_layers': 2,
4. 'units_0': 7,
5. 'units_1': 3,
6. 'units_2': 7,
7. 'units_3': 3,
8. 'units_4': 1,
9. 'units_5': 7,
10. 'tuner/epochs': 25,
11. 'tuner/initial_epoch': 0,
12. 'tuner/bracket': 0,
13. 'tuner/round': 0

This model did not perform as well as the prior two models.



Final results for the fifth model were a loss of 0.556 and an accuracy of 0.728

IV. Summary

In the end, none of the models that I made ended up hitting the target metric of 75% accuracy. I did try a large range of layers and neurons as well as two different activation functions. Considering that these changes did not work, I am led to believe that a better increment in the performance of the model could come from better preprocessing of the input data.
I would also consider running a full test of supervised machine learning algorithms for this problem because I do not think that a deep learning model is optimal for this problem. I would consider an XGBoost, an AdaBoost, or an LGBM model to be the best contenders for this problem. I have seen the best results from those models in the past on other classification problems.

I cannot recommend any of the models that I made in this test because none of them meet the criteria for use given by Alphabet Soup.