# **Overview**

For the nonprofit Alphabet Soup, we created a tool that can help it select the applicants for funding with the best chance of success in their ventures. Using machine learning techniques and neural networks, the features in the provided dataset were used to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

# **Results**:

## Data Preprocessing

* What variable(s) are the target(s) for your model?

The variable that was used as a target for the model was the field 'IS\_SUCCESSFUL'. It is classified as binary 0 and 1 to denote if the money was used effectively.

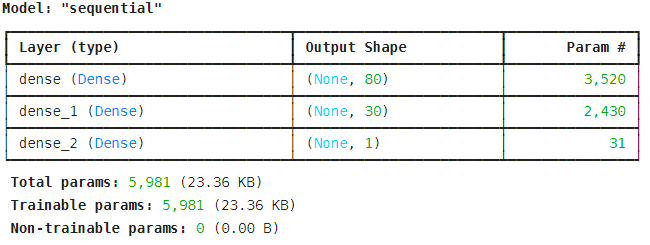
* What variable(s) are the features for your model?
  + ‘APPLICATION\_TYPE’—Alphabet Soup application type
  + ‘AFFILIATION’—Affiliated sector of industry
  + ‘CLASSIFICATION’—Government organization classification
  + ‘USE\_CASE’—Use case for funding
  + ‘ORGANIZATION’—Organization type
  + ‘STATUS’—Active status
  + ‘INCOME\_AMT’—Income classification
  + ‘SPECIAL\_CONSIDERATIONS’—Special considerations for application
  + ‘ASK\_AMT’—Funding amount requested
* What variable(s) should be removed from the input data because they are neither targets nor features?

The variable’s that were removed from the input data were the identification variables ‘EIN’ and ‘NAME’. They were not needed for the model since they do not provide any information that effects the target variable.

## Compiling, Training, and Evaluating the Model

* How many neurons, layers, and activation functions did you select for your neural network model, and why?

For the first model, a total of 2 hidden layers were chosen, the first layer activation being ‘tahn’ and the other two hidden layer activations being ‘relu’. The first hidden layer had 80 neurons and the second had 30. I chose the above architecture as the basis as it provided the most trainable parameters for the model to run. The other models were optimized by changing the activation layers and epochs.



* Were you able to achieve the target model performance?

Although the accuracy of above 75% was not able to be achieved, after optimization, I was able to achieve an accuracy score of 73.59%.

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Base | 0.6817 |
| Optimization 1 | 0.7359\* |
| Optimization 2 | 0.4746 |
| Optimization 3 | 0.7355 |
| Optimization 4 | 0.7352 |

* What steps did you take in your attempts to increase model performance?

The accuracy improved from 68.17% to 73.59% after increasing the epochs from 10 to 55. However, increasing it further from 55 to 100 showed little to no change, decreasing the accuracy to 73.52%. Therefore, leaving the epoch at 55, I ran another model optimization by changing the activation for the first layer to ‘relu’. This however, decreased the accuracy to 47.46%. This emphasized that the activation layer should remain the same and therefore, I tried once again by making the bin size smaller. It once again increased the accuracy to 73.55%. However, all optimizations remained around 73%.

# **Summary**

Overall, the sequential model used with 2 hidden ‘relu’ activation layers and an 80, 30 neurons count with 55 epochs is what provided the best accuracy model. If the hidden layers were decreased and an additional layer were to be added, it may benefit the model accuracy further. From observation it seems that the activation layer ‘relu’ did the best in terms of accuracy but another hidden layer can be changed to ‘tahn’ or even ‘elu’ to improve accuracy.