**Deep Learning Challenge**

**Overview:**

In this analysis we are looking to create a Neural Network model that can predict whether applicants will be successfully funded by Alphabet Soup using a historic dataset.

**Results:**

**Data Preprocessing**

I used "Is Successful" as my target (y) variable for this model, all other columns were used as the features (X) variable.

Fields "EIN" and "Name" were removed from the dataset as they were neither target nor features.

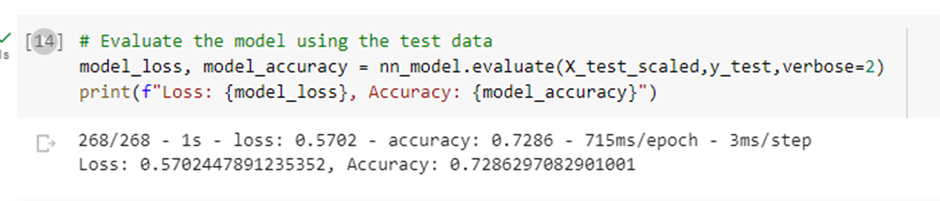
**Compiling, Training, and Evaluating the Model**

Model 1: AlphabetSoupCharity

Using the rule of thumb that for every feature you should use 2-3 neurons for optimal results, I started with 2 hidden layers and below set up for my initial model:

* Layer 1:86, activation "relu"
* Layer 2: 129, activation "relu"
* Final Layer: 1, activation "sigmoid"
* I ran the model for 100 epochs

Based on the above I got a loss of 0.5702, and accurary of 0.7286. The loss rate of 0.5702 means we still have alot of work to do to optimise the model.

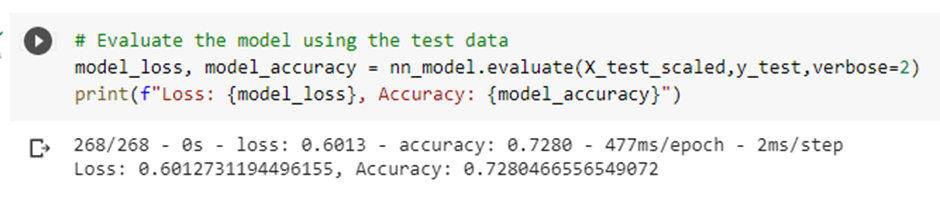
[](https://user-images.githubusercontent.com/116396662/234190897-2eee5ab5-ce07-43b8-87a1-ca47be92a224.png)

Model 2: AlphabetSoupCharity\_Optimisation

For the second model, I tried to reduce the number of columns for "classification" as well as add another layer to the model. Still using the rule of thumb that for every feature you should use 2-3 neurons for optimal results. below was my set up for my second model:

* Layer 1: 80, activation "relu"
* Layer 2: 120, activation "relu"
* Layer 3: 80, activation "relu"
* Layer 4: 120, activation "relu"
* Final Layer: 1, activation "sigmoid"
* I ran the model for 100 epochs

Based on the above I got a loss of 0.6013, and accurary of 0.7280. This had a higher loss rate, however similar accuracy rate to the first model. As the loss rate was higher here, and we still haven't hit the 75% accuracy mark, there is more testing to be done.

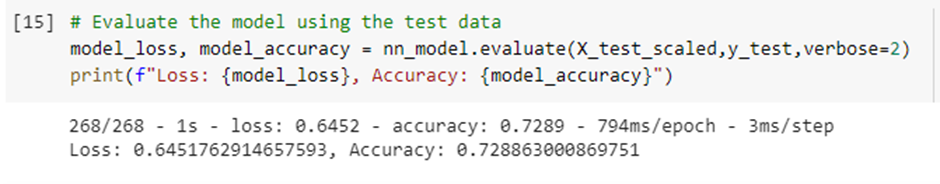
[](https://user-images.githubusercontent.com/116396662/234190928-551b9286-65eb-477f-ba16-5f45ea4b70c0.png)

Model 3: AlphabetSoupCharity\_Optimisation2

For the third model, I tried to switch things up a bit given going by the rule of thumb advice doesn't seem to be working for this analysis. Below wwas the set up for my final model:

* Layer 1: 120, activation "relu"
* Layer 2: 240, activation "relu"
* Layer 3: 240, activation "relu"
* Layer 4: 150, activation "relu"
* Final Layer: 1, activation "sigmoid"
* I ran the model for 100 epochs

Based on the above I got a loss of 0.6452 and accuracy of 0.7289. This model had the most lost so far with only a slight accuracy improvement.

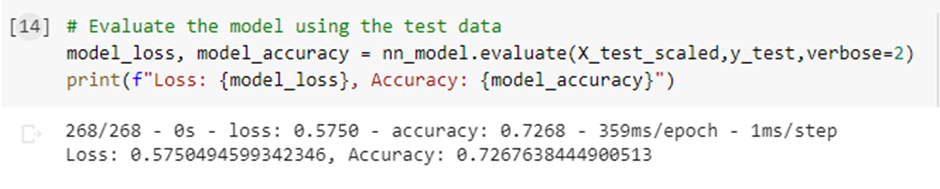
[](https://user-images.githubusercontent.com/116396662/234190951-c71edaf6-713f-4548-b1be-d7188ed7ecc0.png)

Final Model: AlphabetSoupCharity\_Optimisation3

For the final model, I tried to a new approach given changing neurons and layers only hasn't worked so far. For this model I removed additional columns USE\_CASE, ORGANIZATION AND SPECIAL\_CONSIDERATIONS to try reduce the number of features. With the reduced number of columns, I then reduced my neurons in the layers to follow the 2-3 neurons per feature rule of thumb. Below was the set up for my final model:

* Layer 1: 64, activation "relu"
* Layer 2: 96, activation "relu"
* Layer 3: 64, activation "relu"
* Final Layer: 1, activation "sigmoid"
* I ran the model for 100 epochs

Based on the above I got a loss of 0.5750 and accuracy of 0.7268. The reduction of features did little to improve the model and had similar results to the original model.

[](https://user-images.githubusercontent.com/116396662/234190978-26604d71-6015-4e26-8de3-0c6ccc386a9a.png)

**Summary:**

It seems this is a tough dataset to crack. It likely needs alot of layers to split out the data. You could potentially leave the dataset as is without removing the initial columns requested and/or use the Keras Tuner Function to run through multiple options which may help you to find an optimal model.