Alphabet Soup Model Performance

# **Overview**

The non-profit foundation, Alphabet Soup, want to create a tool that will predict whether applicants will be successful in their ventures. Their current dataset contains over 34,000 records of all organisations that have previously received funding from Alphabet Soup, this includes whether or not the venture was successful.

# **Results**

## Data Pre-processing

In total, there are 12 variables in the input dataset. The target for the model is the ‘IS\_SUCCESSFUL’ variable. The remaining variables are classed as features; however, both the ‘EIN’ and ‘NAME’ columns were dropped on the basis that they would not be beneficial to the model as they only contain identification data. Therefore, there are 9 feature variables.

Text

Description automatically generatedText

Description automatically generatedIn addition to the above, the variables ‘APPLICATION\_TYPE’ and ‘CLASSIFICATION’ were binned to group together “rare” values into a value ‘Other’. All categorical variables were then encoded using pd.get\_dummies().

## Compiling, Training, Evaluating the Model

Text

Description automatically generatedIn the initial neural network model, there were 3 layers in total (2 hidden layers and the output layer). Both hidden layers had 16 neurons each and used the ReLu activation function, while the output layer used the Sigmoid activation function.

After testing ReLu against Tanh, ReLu demonstrated a higher accuracy and so was chosen for this model.

The Sigmoid function was used for the output as the target that is required is either a 0 or 1. As Sigmoid outputs a single probability of belonging to one class, it is best suited for binary classification.

The initial model produced an accuracy of 72%.

Further discussion on the number of neurons and hidden layers chosen is in the section below.

### **Model Optimisation**

Text

Description automatically generatedIn the process of model optimisation, I trialled different methods. One of which consisted of increasing the number of neurons in the hidden layers. This slightly improved the accuracy, however not by much. I eventually chose 30 neurons in each layer as increasing this further had very little positive impact on accuracy.

Another method I trialled was to add a third hidden layer, however this had very little impact on the results and, in fact, worsened the accuracy score. This could have been caused by overfitting on the training data.

I found that reducing the cut-off figures for the bins for both the ‘APPLICATION\_TYPE’ and ‘CLASSIFICATION’ variables improved the score. However, this still did not reach 75% accuracy. Also, removing any other variable from the dataset worsened the accuracy.

Text

Description automatically generatedThe model did obtain over 75% accuracy when I re-introduced the ‘NAME’ variable. As there are multiple ventures carried out by the same applicant, this could help identify the success outcome if that same applicant applies again. I created a bin for the rare occurring applicants and ran this through the model. However, as this variable is an identifier, it may cause confusion to the model if a new applicant applies that the model has not previously seen.



# **Summary**

Although I managed to reach over 75% accuracy, this is still not a very good model. In addition, the ‘NAME’ variable that I re-introduced could cause issues on new data where it would not add any value to the model, therefore is probably better to remove altogether from the dataset.