**Performance Review – Alphabet Soup Application Success Predictor**

1. **Overview**

A deep learning model has been created for the nonprofit foundation Alphabet Soup. The purpose of this model is to predict whether a funding application is likely to be successful or unsuccessful. However, the model needs to be accurate and trustworthy once generalised and therefore a reviewal of the steps taken to create the model and it’s performance will be undertaken to help create transparency and evaluate the model created.

1. **Results**:

Data Preprocessing

* **Target variable:** IS\_SUCCESSFUL – a binary classification of if an application is likely to result in the money being used effectively [successful (1) or unsuccessful (0)].
* **Features:** The variables which are fed into the model to aid prediction:
  + APPLICATION\_TYPE – The type of Alphabet Soup application
  + AFFILIATION – The affiliated sector of industry
  + CLASSIFICATION – Government organisation classification
  + USE\_CASE – Use case for funding
  + ORGANIZATION – Organization type
  + INCOME\_AMT – Income classification
  + SPECIAL CONSIDERATION – Special consideration for the application
  + ASK\_AMT – Funding amount requested
* **Removed variables:** Removed from the input data because they were not considered to be either a target or a feature within the optimized model:
  + EIN – Identification
  + NAME – Identification
  + STATUS – Active status

Compiling, Training, and Evaluating the Model

* **Neurons:** 11 neurons were used in the first hidden layer and 28 neurons were used in the second hidden layer.
  + A general rule states that 2-3 x more neutrons than the input features should be used. However, with 42 input features, it was preferable to avoid this rule whilst still achieving the desired accuracy as more computational resources would be required.
  + A keras auto tuner was ran which trialled the possible hyperparameter combinations on the original model created. The suggested number of neurons in the results were used as guidance which returned a near respectable accuracy.
  + However, some changes were made to composition of the features within the optimized model, and due to time constraints another keras autotuner was not ran. Therefore, to help stem efficiency on the optimised model, the number of neurons was increased in the second layer to help stem more communications between the computations which were taking place on each node due to the complexity of the input features.
* **Layers:** 3layers were used
  + There is an ongoing argument within the data scientist community that even the most complex interactions can be characterized by as few a 3 hidden layers.
  + 3 was sufficient in considering enough of the interactions between variables. Although some of the Keras autotuner results suggested to use more than 3 layers, with each additional layer comes more computational resources and therefore for a non-profit organization, 3 was adequate.
* **Activation Functions:** Layer 1 - Tanh, Layer 2 – Relu, Layer 3 - Sigmoid
  + Layer 1 – Tanh: The Tanh activation function was used as it can be used for classification or regression. It was able to take in the input features and transform the output to a range between -1 and 1. On the initial data set, it helps to classify data into one of two classes which would then be fed into the next layer where more complex patterns could be distinguished.
  + Layer 2 – Relu: The rectified linear unit activation function was used as it is ideal for modelling non-linear input data for classification or regression. All the -1 values from the first layer would be changed to 0 at this stage which still yielded a accuracy of 75% so this combination seemed to work well with the input features.
  + Layer 3 – Sigmoid: Generally, a less complex activation should be used on the output layer as the output is going to take the form of a simple binary classification. By this layer, only the most meaningful synapse communications would reach this layer. Therefore, the sigmoid activation function was selected due to its ability to simply predict probabilities and using them to categorise an s-curve.
* **Model Performance:**
  + Once optimised, the 75% target performance was met.
  + The initial model did not reach this 75% accuracy bench mark on the testing data, it instead returned an accuracy of 72.6%. Therefore the following steps were taken to increase model performance:
    - On the original model, STATUS was considered as a feature. However, this column was dropped upon optimization to reduce noise as the machine learning algorithm appeared to be learning bad habits from this feature.
    - The number of values in the ‘other’ bin was increased for the CLASSIFICATION feature. On the original model, any values which represented less than 1% of all applications were binned into an ‘Other’ bin. However, this was increased to 3% in the optimized model.
    - The unoptimized model has 21 neurons in the second hidden layer, this was increased to 28 in the optimized model.
    - Within the unoptimized model, the first layer has an activation of relu, however, this was changed to tanh in the first layer of the optimized model.
    - The number of epochs was deliberately not increased due to the amount of input features, if the epochs was significantly increased, it ran the risk of overfitting.

1. **Summary**: Summarize the overall results of the deep learning model. Include a recommendation for how a different model could solve this classification problem, and then explain your recommendation.