**DEEP LEARNING HOMEWORK**

**TOPIC: Charity Funding Predictor**

**Overview**

This project work seeks to use binary classifier that will be capable of predicting whether applicants will be successful if funded by Alphabet Soup. In the quest for better performance and a business case, increasingly complex methods are being used to model patterns to achieve success. The realism, however provided by these models comes at a cost of greater computational complexity. This work report explores the use of neural networks to help shape the organization going forward.

**Data Processing**

Data was provided in csv format for this work:

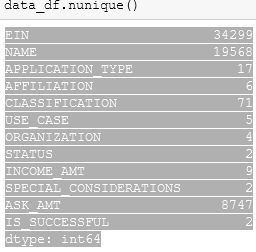
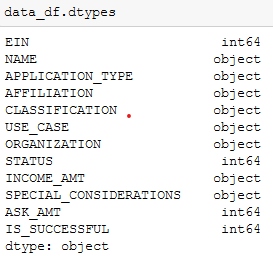
In order to process and work on the data, a Jupyter Notebook was combined with Google Colab since most of the libraries were not adaptable to my computer in this case. However, screenshots and tables were downloaded to give more insight and clarity.

**Libraries:**

* Pandas to read and create data frames
* Seaborn for visualization
* Sklearn for training and testing and standardizing data
* Tensor flow for machine learning
* Keras tuner for hyper parameter optimization
* Keras layer dense to create sequential model

**Data cleaning:**

The raw data consists of 34299 names with 12 features in columns. A look at the datatypes and unique columns as shown below gives an idea as to how to clean and process the data.

  
**Data target variable**: IS\_SUCCESSFUL

**Data feature(s) variables**: APPLICATION\_TYPE, AFFILIATION, CLASSIFICATION, USE\_CASE, ORGANISATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATION, and ASK\_AMT.

**Irrelevant Variable(s):** EIN, NAME.

**Data Compilation, Training and Evaluation:**

For effective neural network training, the following processes was needed:

1. Remove irrelevant columns
2. Adopt the split function to separate the income amount and also define a function to convert rows with M to represent a million.
3. Reduce the number of unique columns in this case:
   1. Classification
   2. Application Type
   3. Create an OneHotEncoder instance using the categorical variable list and then add to the DataFrame.
   4. Merge encoded features and drop the originals.
   5. Split the processed data into features and target arrays
4. Data scaling for effective neural neatwork training.
5. Apply train\_test\_split function for model training
6. Create model function
   1. Define the NN model
   2. Add hidden layers using relu
   3. Add output layer using sigmoid function
   4. Check model
   5. Evaluate model

**Results:**

The summary of the results for the neutral network run on all the data set are shown fig.1 below.

**Model1:**

Assumptions

1. Reduce unique categories of Application type to 10 by creating a bin
2. Reduce categories column to less than 800

**Model2:**

Assumptions

1. Reduce unique categories on Application type further to those less than 5 by creating a bin
2. Slash classification to less than 1800
3. Filter names with income amount greater than or equal to 10000.

**Model3:**

Assumption

Leave all features and categories as describe in dataset.

**Model4:**

Similar to model 1 except increase the number of neurons and reduce epochs

**Activation function used:**

1. Relu
2. Sigmoid
3. Softmax



The accuracy and loss scores on each neutral network are listed for the models developed. In particular one can notice that it is difficult to obtain 75% of the target model performance as our optimization only produced the best performance of approximately 73%.

**Recommendation:**

Since by design the ReLU activation function is unbounded in the positive domain. A further reading indicates that some form of weight regularization will help prevent potential numerical problems which will intern promote additional sparsity. Also, since the features in this project work are quite complex and not very much related the sigmoid function even though produced an accuracy of 73.2% it cannot learn complex mapping hence its deficiency for this model. The use of Convolutional Neural Networks (CNN) might be appropriate.

**Challenges:**

1. In running automation of the hyper parameters and trying to find the best tuner for the model since this function was always producing errors.
2. Having challenges running TensorFlow on local machine and most of the job done in google Colab. Hence the only way is to download and save or screenshot for visualization.