**Module 19 Challenge: Alphabet Soup Charity: (Machine Learning Neural Networks)**

**Challenge Overview:**

This week’s challenge was to use the TensorFlow library to create a “neural network" machine learning model that could be used to evaluate funding request applications for charitable projects to predict which projects if funded would be successful or not. The expectation was for the model to have an accuracy rate above 75%.

We were provided with a dataset that contained historical data on previous charitable projects that were funded. Below is a list of the data provided in the dataset:

* **EIN** and **NAME**—Identification columns
* **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 consideration for application
* **ASK\_AMT**—Funding amount requested
* **IS\_SUCCESSFUL**—Was the money used effectively

Using the Machine learning process:

PCA – clean and standardize the data

PCA – remove unnecessary data features

Develop a TensorFlow neural network

Compile, Train and Evaluate for accuracy

**Process and Results:**

* **Data Pre-processing**:
  + Determine what variable is the targeted result for the model.
    - We want to determine if a funded project will be successful or not, so the **IS\_SUCCESSFUL** column will be the target variable.
  + Determine what variable should be kept as features for the model.
    - It was determined that all non- Identification columns were independent variables that could help predict our desired target.
  + Determine what variables should be removed.
    - Removed the identification columns: (EIN & Name)
* **Develop, Compiling, Training, and Evaluating the Model**:
  + Creating the neural network, number of Neuron layers and activation functions:
    - The following were given in the challenge for use in the original model:
      * 2 hidden layers, with 80 and 30 neurons respectively
      * 1 output layer
      * 2 different activation functions, relu and sigmoid
    - To optimize the model and try to improve its accuracy various changes were tried, changing the neuron splits, changing the number of hidden layers, and using different activation functions. The following seemed to be the best for efficiency, yet it still did not have the required accuracy level:
      * 3 hidden layers, with 100, 50, and 20 neurons respectively
      * 1 output layer
      * 3 different types of activation functions, relu, tanh, and sigmoid
  + Achieving accuracy levels greater than the required 75%
    - **The** model was successful in meeting the required accuracy rates of greater than 75%. Ultimately, its accuracy was around 80%

**(Below is the results of the Optimized code)**

Graphical user interface, text, application

Description automatically generated

* + What steps did you take to try and increase model performance?
    - Several changes to the model were tried by “trial and error” to achieve the required accuracy level:
      * First, in the optimization process the number of hidden layers and neuron splits were changed and different activation functions were tried with minimal effect on the accuracy.
      * Second, the bucketing of data was reviewed and modified to see if that would have an impact on the resulting accuracy. Originally the INCOME\_AMT was “binned” into nine buckets. That was changed to just two buckets, yes it had income, or no it did not have income, this had little effect on the accuracy.
      * Finally, in reviewing all the data provided to see if there could be something else that would impact the results of the model. It was determined that several organizations had multiple funding request for different projects, so if previous projects from this organization were successful it would be a good indicator that any future project would also be successful. So, the names of the requesting organizations were added back into the model and binned into those with one application verses those with more than one. This final change to the model allowed it to achieve the required accuracy levels.

## Challenge - Summary

While the model was finally able to achieve the required objective of an accuracy level greater than 75% it took a lot of effort and time to get there. Since I didn’t have anything that would give me a hint to what changes would do to the model, all I could do was trial and error. That is make a change and then run the code to see the results, if it was not what I wanted then I would try something else. Perhaps that is true to form in the real world, but I found it rather bothersome. I would suggest that rather than using a neuron network model perhaps one of the other machine learning methods would be better, such as the LogisticRegression model or RandomForestClassifier, these both had methods to narrow down what changes you should try to make improvements. It might be a good idea to try several different model types and compare their results to see how similar their results might be.