**AlphabetSoup Campaign Success Modeling Report – Neural Network Modeling and Prediction Success Rate**

**Overview:**

In response to the request from AlphabetSoup to investigate and create a machine learning model using deep neural networks to predict if an applicants’ campaigns will be successful if funded by AlphabetSoup.

AlphabetSoup has provided historical data of ~34,000 applicants/organizations that received funding and if the campaign was “Successful”. This data will be leveraged though various processing, modeling and optimization tools to drive to the most accurate modeling of success rates possible with a goal of 75% or more accuracy. The data, tools and outcomes are captured below within the Results and Summary details.

**Summary:**

Using Neural Network modeling and leveraging several different optimization efforts, predictability of the success was improved slightly from 72.5 to 72.6% which is not meaningful. Using feature and column reductions and different functions to drive the underlying Neural calculations clearly demonstrated improvement. Final accuracy is still considered to be less than the desired 75%. Detail results are captured in the report below.

Additional modeling actions could be leveraged if AlphabetSoup would like to explore the following:

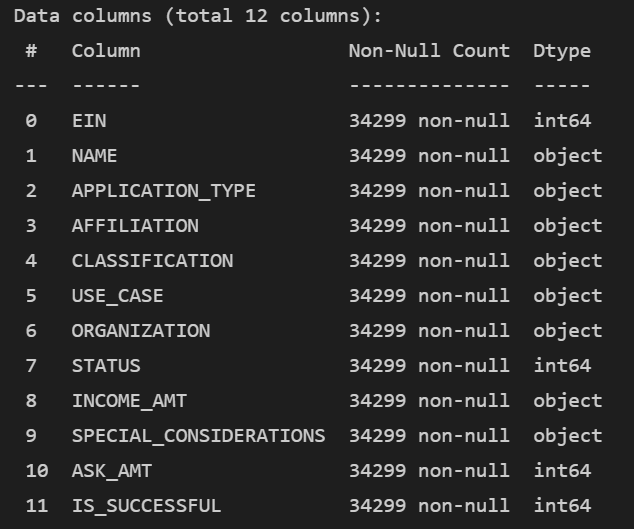
1. Increasing the tuning of the Neural Network Model through the usage of pipeline tuning.
2. Automated dimension reductions using PCA or tSNE methods vs. manual methods used in this evaluation could increase accuracy of any model, Neural or otherwise.
3. Also, the usage of clustering models vs. Neural Network Modeling may also provide increased accuracy of predictions.

**Results:**

**Data Preprocessing:**

* Initial review of the data provided indicated that the target for the model was if the applicant/campaign successful, using the “IS\_SUCCESSFUL” column (see Fig. 1).
* Two variables or data points in the data set were identified as not being meaningful to the model outcome, “EIN” and “NAME”; these are strictly identifying names/identification numbers of the applicant (see Fig. 1).
* The remaining features in the data set were used to in the model (see Fig. 1):
  + 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

**Fig. 1** - Data Columns and Data Types form the AlphabetSoup Historical Data:



After the target variable were identified and the identification columns (EIN/NAME) were removed, the unique counts of each data variable were reviewed to identify variables that have 10 or more unique values and further determining the number of each of the unique variable types (see Fig. 2).

Fig2 – Unique Feature types by Feature:

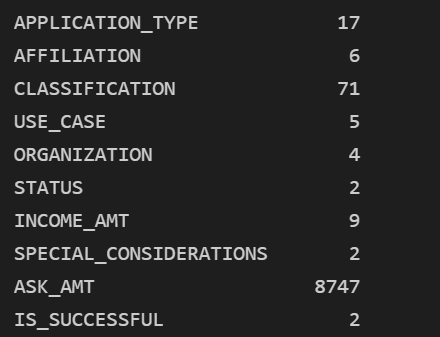
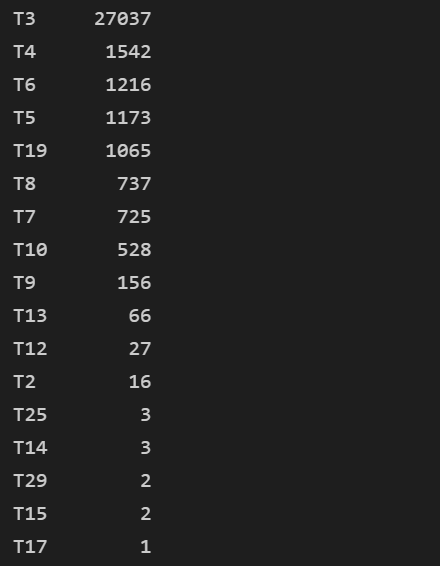
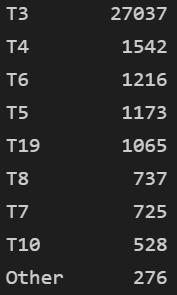


Fig. 3 – Unique Application Types:



The application type was reviewed with 17 application types. Any type with a count less than 500 our approximately half of the application types but very view of the actual applicants were in these types. These applicants were all grouped as “other” to eliminate potential errors our impacts form these outliers or dilution.

Fig. 4 – Unique Application Types after Binning



Next the Classification feature was reviewed for potential feature reduction for the same reasons as above.

Fig. 5 – Unique Classification counts > 1 (71 unique classification exist)

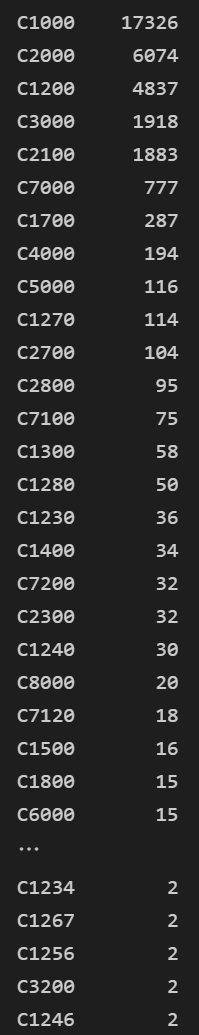
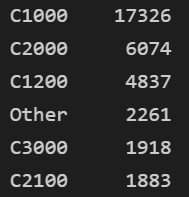


Fig 6. – Unique Classification Counts after Binning

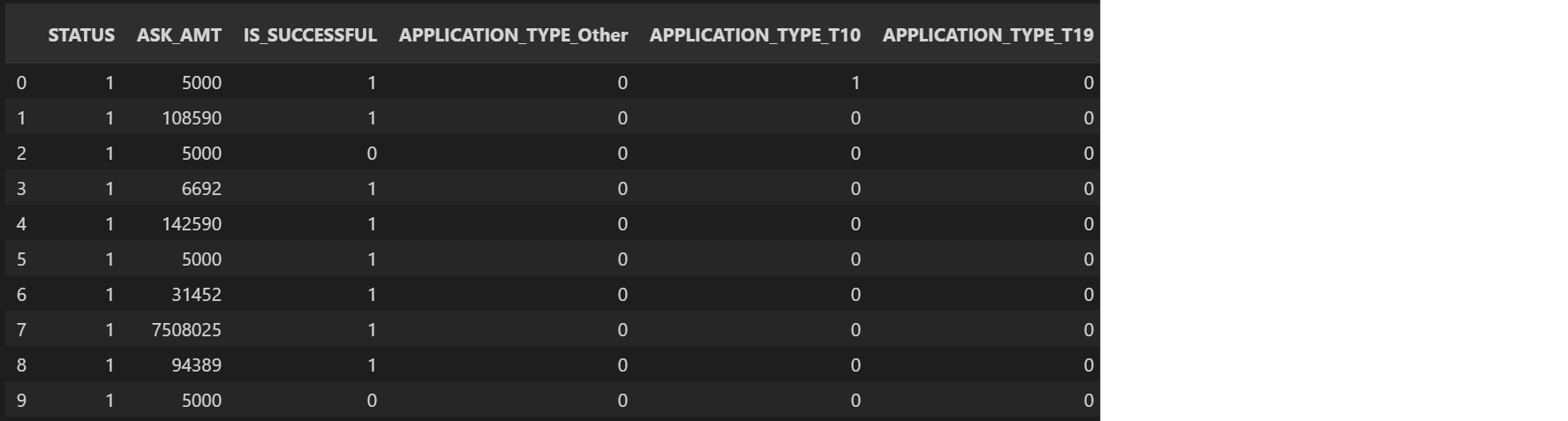


By leveraging binning, the classification feature has been reduced from 71 classifications to 6 as seen in Fig. 6.

While the Ask Amount for each applicant is a feature with greater than 10 unique values, it was not binned for the initial modelling as this may be viewed as a key feature in determining outcome.

To continue the data pre-processing from the data provided by AlphabetSoup, a programing feature called “GetDummies” was utilized to convert all categorical data to numeric data. This step is crucial in the process. Machine learning models require numeric data explicitly to predict outcomes.

Fig. 7 – Example of Categorical data converted to Numerical Data

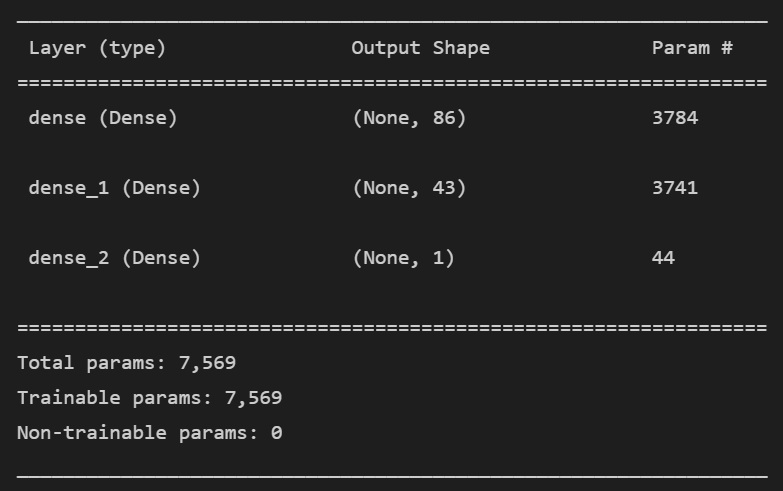


The last steps in the data pre preprocessing of the data is to split the data into testing and training sets for training and testing of the model to determine the accuracy using the data as process. Additionally, the data was scaled. Scaling of data is a key step in assuring that large variance in data feature values does not adversely influence the model by making all values fall within a similar scale.

**Compiling, Training and Evaluating the model:**

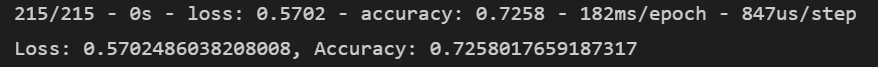
A neural network model uses a predetermined number of nodes or layers to evaluate or model outcomes based on the historical training data. Two input layers were leveraged in the initial model. The first input layer is using 86 nodes or twice the input features and the second node uses the number of features or 43. Lastly the number of training cycles or “Epochs” is set to 100. The neural network will begin training and weighting each feature within the epochs to determine which parameters across the features will predict more accurately the target outcome of the success of the campaign.

Fig. 8 – Initial parameters the model with train itself with:



As the Model Runs, each Epoch will determine the loss and accuracy of the parameters in the epoch to assist in defining the most accurate model. Lower Loss and Higher Accuracy is always the intended target.

Fig 9. – Initial Model Accuracy before optimization:



The initial model demonstrated .57 loss and .72 accuracy after the initial training. This is short of our goal of .75 or 75% accuracy in predicting the success of an applicant/campaign funded by Alphabet Soup.

**Model Optimization:**

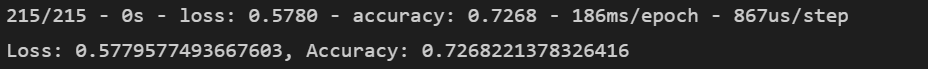
**First Optimization** – Increasing Training Epochs from 100 to 200 for model training (see Fig. 10).

Fig. 10 – Epochs = 200



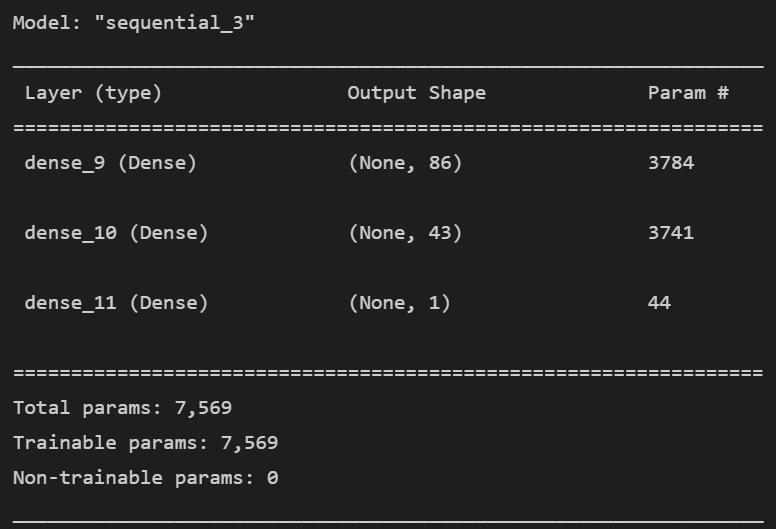
Increasing the Epochs resulted in slightly increased accuracy but also slightly increased loss (see Fig. 11)

Fig. 11 – Increased Training Epochs Model Loss and Accuracy:



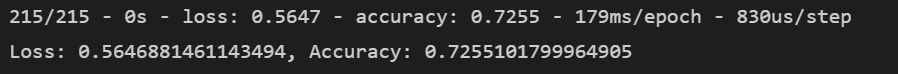
**Second Optimization** – Reverting back to 100 training Epochs; replacing the ReLU activation function in the model with the Tanh activation function. This function will use a more complex formula to predict accuracy, which may cost more time/or loss to complete. The neural nodes and input features have not been changed. The resulting parameters are the same as the original modelling criteria (see Fig. 12).

Fig. 12 – Tanh Activation Optimization Parameters:



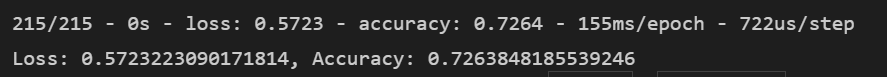
Using the Tanh function with the same neural layers and features resulted in both increased loss and lessor accuracy when using 100 Epochs.

Fig. 13 – Tanh Activation Function Model Loss and Accuracy:



Using the Tanh function with the same neural layers and features and increasing the training epochs resulted in loss of .57 and accuracy of .726 (see Fig. 14). It would appear that additional epochs have benefited accuracy in both the activation functions used (ReLU vs. Tanh).

Fig. 14 – Tanh Activation Model Loss and Accuracy at 200 Epochs:



**Third Optimization** – Additional Feature dimension reduction is being utilized to created additional bins for data with larger counts of unique values and also reducing redundant columns.

Income amounts were reduced from 9 different bins (see Fig. 15) to 4 combining any income amounts below $3000 into one category and binning the remaining categories (see Fig. 16).

Fig. 15 – Initial Income Bins

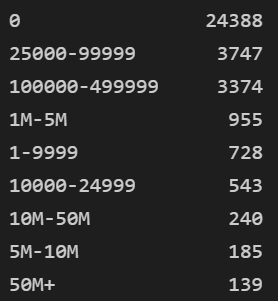
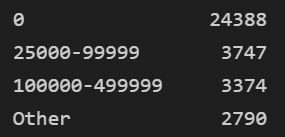


Fig. 16 – Reduced Income Bins



Next the unique values in the affiliation counts were also reduced from 6 (see Fig. 17) to 3 (see Fig. 18).

Fig. 17 – Initial Affiliation Counts

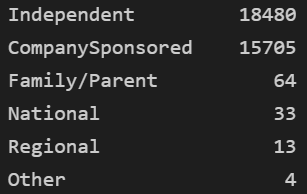
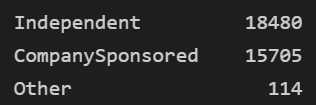


Fig. 18 – Final Affiliation Counts



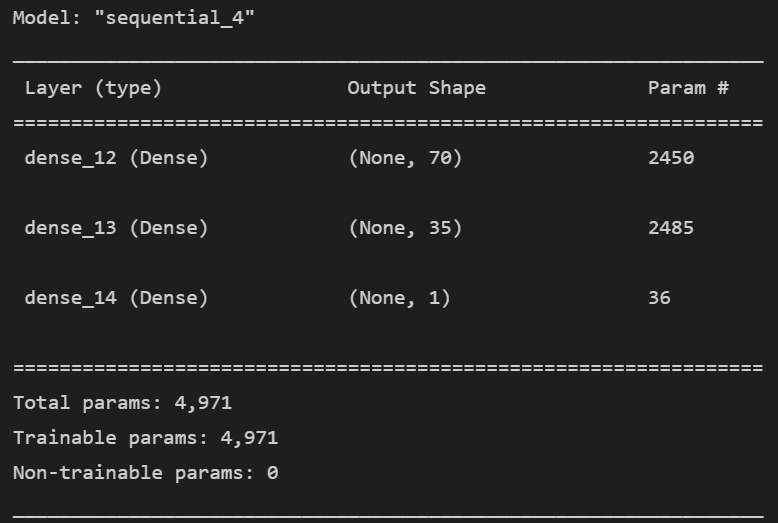
After reducing the unique features in the dataset for Income and Affiliations, the categorical data was once again converted to numeric values for usage in the neural model.

Before modeling, one additional feature reduction was applied. When the categorical data is converted to numeric data, two columns are created for the “Special\_Considerations” column, Y or N. This is duplicative because the model only needs the Y values to consider this in the activation function computations.

After reducing the features in the dataset, the same data split to training and testing data was completed and the scaling of data was completed. These activities were completed identically to the original model.

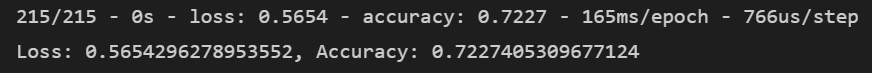
**Fourth Optimization** – Feature Binning and Reduction with reduced neurons using the same model activation and epochs as the original model. By reducing the features and columns, the parameters are reduced to 4,971 compared to ~7,000 in the original modeling (see Fig. 19).

Fig. 19 – Reduced Training Parameters in Fourth Optimization



The model was trained over 100 Epochs on the reduced data set and when applied to the test data set the demonstrated a loss of .56 and accuracy of .722 which is better loss but less accurate than the original model.

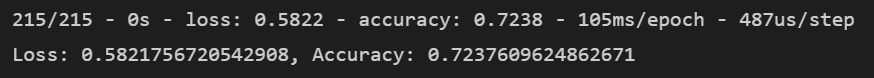
Fig. 20 – Fourth Optimization Results:



**Fifth Optimization** – Reduced Training Parameters with 200 Epochs

Using 200 Epoch on the reduced data set increased loss and slightly increased Accuracy to .723 (see Fig. 21).

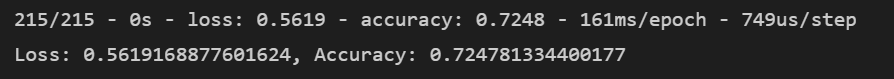
Fig. 21 – Fifth Optimization Results:



**Sixth Optimization** – Reduced Training features using the Tanh activation function and 100 Epochs.

When reducing data features and columns and using the tanh activation function loss was observed at .561 and accuracy of .724 which is improved loss over the original model and slightly less accurate than the original but improved over both compared the “ReLU” activation function.

Fig. 22 – Sixth Optimization Results:



**Seventh Optimization** – Reduced Training Features using the Tanh Activation function and 50 Epochs.

When reducing the data features and columns and using the Tanh activation function with increased training epochs, loss was observed at .564 and an accuracy was observed at .726 (see Fig. 23). which is minorly better that he original model and slightly better than most optimizations.

Fig. 23 – Seventh Optimization Results:

