

[This Photo](https://sefiks.com/2017/01/28/backpropagation-implementation-from-theory-to-action/) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)

|  |
| --- |
| Module 21 Challenge:  Deep Learning |
|  |
| July 8, 2024  Authored by: Justin Bein |



# Week 21 Challenge: Deep Learning

## Background[[1]](#footnote-1)

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With your knowledge of machine learning and neural networks, you’ll use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

From Alphabet Soup’s business team, you have received a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. Within this dataset are a number of columns that capture metadata about each organization, such as:

* 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 considerations for application
* ASK\_AMT—Funding amount requested
* IS\_SUCCESSFUL—Was the money used effectively

## Overview

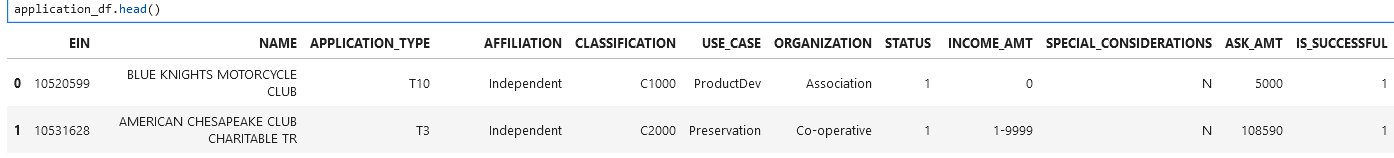
The purpose of this analysis was twofold: first, the overarching objective was to build a deep learning neural network to automatically evaluate applications for funding. Second, model optimization was undertaken in an attempt to refine the approach. That is, by employing various approaches, e.g., wrangling the data; adding layers; adding nodes; trying different activation functions, it is hoped that the accuracy of the model can be improved. It is important to note that improvement in the model should not come at the expense of overfitting the model or causing confusion in the model.

## Results

### Data Preprocessing

Before building the model, exploratory data analysis and preprocessing was performed. My approach:

1. The various dependencies (libraries, etc.) were imported, including:
   1. train\_test\_split
   2. StandardScaler
   3. Pandas
   4. Tensorflow
   5. ModelCheckpoint
2. After importing the dependencies, the dataset (CSV) was imported and read
3. An excerpt of the dataset was viewed to familiarize myself with the data types and values:

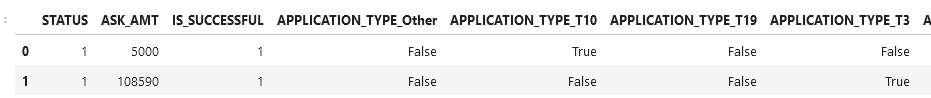


After familiarizing myself with the data, I answered the following questions:

* What variable(s) are the target(s) for the model? “IS\_SUCCESSFUL” is the target. Values of “1” indicate the project was successful, whereas “0” indicates the project was not successful. As an aside, “success” wasn’t defined.
* What variable(s) are the features for the model? At the outset of building the model, I the following variables were deemed features:
  + APPLICATION\_TYPE
  + CLASSIFICATION
  + USE\_CASE
  + ORGANIZATION
  + STATUS
  + INCOME\_AMT
  + SPECIAL\_CONSIDERATIONS
  + ASK\_AMT
* What variable(s) should be removed from the input data because they are neither targets nor features? At the outset of building the model, the following variables were removed from the input data:
  + EIN
  + NAME

There’s intuitive appeal to dropping these variables – the EIN is, loosely speaking, a numerical representation of the applicant’s name. Also, an applicant’s name should have no bearing on the success of a project. An argument could be made that the name does have an impact, particularly if the name is used in advertising. For example, an applicant name of “Fraudsters, Inc.” doesn’t instill confidence in those providing funding…

1. I then examined the number of unique values in each column, and where appropriate, binned values to reduce the dimension of the data. This was done to avoid confusing the model. For example, there were 71 unique classifications and many of these had only 1 occurrence. This is indicative of unbalanced data. It was appropriate then to reduce the 71 to 6.
2. Categorical data was converted to numerical data via “pd.get\_dummies”. This is a standard approach and is done so the neural network can evaluate the categorical data. For example, each record had an “APPLICATION\_TYPE”, and after binning (step 4), there were 9 types. For each record the “APPLICATION\_TYPE” was evaluated at “true” or “false”:



In the snippet above, record 0 has APPLICATION\_TYPE = T10.

1. After binning and converting the data, the dataset was split into two arrays: the target array and the features array.
2. The arrays were then split into training and testing data.
3. Lastly, the training and testing data was scaled. Scaling was done to ensure that no column has overtly high influence solely due to the nature of the values within the column. Note: Scaling is analogous to computing Z-scores

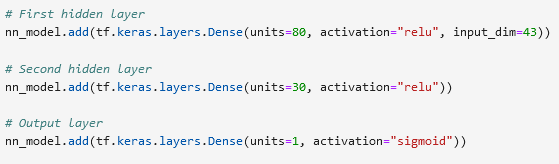
This concluded the preprocessing.

### Compiling, Training, and Evaluating the Model

I then build the model using tensorflow and the keras API for python:



The model contained the inputs (obviously!) two hidden layers, and the output layer (obviously!):



## The structure of the model:

Summary of the model:

Layer 1: 80 neurons, activation function = “relu”

Layer 2: 30 neurons, activation function: = “relu”

Layer 3: 1 neuron, activation function = “sigmoid”

The number of neurons selected for the first and second hidden layers were suggested by starter code. I tried various combinations of these but discarded them because the model results weren’t materially different – and more importantly, didn’t improve the model’s predictive ability.

ReLU (Rectified Linear Unit) is defined as f(x) = max(0,x), where x is the input to the function. I used this because, according the course instructor, ReLU is computationally efficient and helps alleviate the vanishing gradient problem.

The Sigmoid function is defined as f(x) = 1/(1+ exp(-x)), where x is the input to the function. The output of the Sigmoid function is always in the range of (0,1), and therefore it is suitable for binary classification problems. Per the Xpert Learning Assistant: “[…] the Sigmoid function is commonly used in the output layer of binary classification models where the goal is to predict probabilities of class membership.”

This model yielded the following results:

Loss: 0.5604

Accuracy: 0.7294

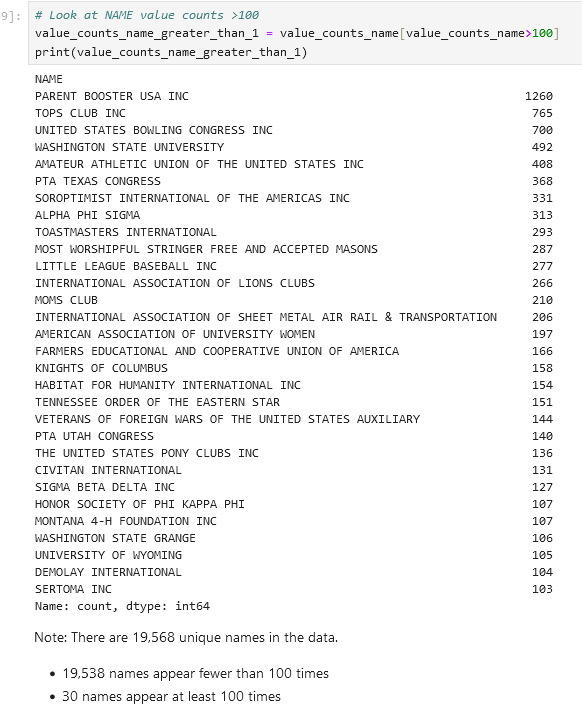
This means the model achieved an accuracy of approximately 72.94% on the test data. This isn’t encouraging. The loss value, which indicates how well the model is performing, was 0.5604. Note: A lower loss value is better.

Conclusions:

1. The target accuracy of 75% was not reached.
2. This model performed moderately well, but there’s room for improvement.

After obtaining the initial results, I attempted to optimize the model; that is, improve the accuracy. To this end, I evaluated one of the variables I had dropped initially: NAME. A quick analysis of this variable indicated there were 19,568 unique names for the 34,299 EINs. This suggested to me that 1) including NAME; and 2) binning this variable could improve the model.

The results of my exploratory data analysis of NAME:



In addition to the binning that was done for the first model, I binned the names based on the number of occurrences, e.g., NAME with less than 100 occurrences were binned as “OTHER”.

The remaining steps for this model were the same as those for the previous model.

This model yielded the following results:

Loss: 0.5068

Accuracy: 0.7523

This means the model achieved an accuracy of approximately 75.23% on the test data. This is encouraging. The loss value improved as well: 0.5068.

Conclusions:

1. The target accuracy of 75% was reached.
2. This model performed well; however, it would be time well spent to attempt the “optimizer” functionality and see if other combinations of layers, nodes, and activations functions would result in greater accuracy.

## Summary

Both models performed fairly well, with the second, more detailed (in the sense that more input variables were use) having the better results.

Options to consider for improved performance: Resources permitting, it would be worthwhile to try models with the following:

1. Expand the binning for each of the three variable categories that were binned. It could be the case that there’s an “inflection point”, e.g., 50 or more occurrences rather than 100, that would aid in the prediction of the outcomes.
2. Adding a third, hidden layer. As I understand it, an additional layer could permit more results from the previous layer to be evaluated. It could be the case that some of the results from the current first layer do not have much influence. By adding a third layer, it is possible that more results could be incorporated.

I am disinclined to add more neurons to the model – there are already numerous neurons, and adding more doesn’t seem to have much lift.

## Source Data

"https://static.bc-edx.com/data/dl-1-2/m21/lms/starter/charity\_data.csv"

1. From Module 21 Challenge, https://bootcampspot.instructure.com/courses/5095/assignments/72083?module\_item\_id=1192121, accessed 07/07/2024 [↑](#footnote-ref-1)