|  |  |  |  |
| --- | --- | --- | --- |
|  | Jeff Pinegar | | jeffpinegar1@gmail.com |
| Assignment: | Deep learning Challenge | 717-982-0516 |
| Due Date: | March 6, 2023 |  |

**Deep Learning Trials**

# Overview

Investigation of

The non-profit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. 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 several columns that capture metadata about each organization.

* 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

## Objective

Build a model with at least 75% accuracy for predicting success.

## Results

Despite using three different data sets, two different auto-tuning methods, more than 2,000 model fits, 2 to 6 dense layers, and multiple activation functions, I could not move above about 73%. In very few cases did the accuracy fall below 71%. The results were “trapped” in a very narrow and surprising range.

Below are the details of my investigation.

Contents

[Overview 1](#_Toc128932278)

[Objective 1](#_Toc128932279)

[Results 1](#_Toc128932280)

[Methodology 2](#_Toc128932281)

[Data Set Preparations 3](#_Toc128932282)

[ASK\_AMT Scaling 4](#_Toc128932283)

[Search Methods and Number of Layers 5](#_Toc128932284)

[Activation Functions 5](#_Toc128932285)

[Results 6](#_Toc128932286)

[Full Data Set 6](#_Toc128932287)

[Typical Results 7](#_Toc128932288)

[Program files 8](#_Toc128932289)

[Model Results 8](#_Toc128932290)

[Half Data Set 8](#_Toc128932291)

[Random Forest 9](#_Toc128932292)

# Methodology

In the search for a model with accuracy above 75% I tried several data preparations, numbers of layers, activation functions, and search methods; Table 1, summaries those attempts.

Table : Attempted Models



## Data Set Preparations

This describes the clean of the data except for ASK\_AMT. ASK\_AMT is the only numeric independent variable, so it required special treatment discussed separately

* Data Set One (Minimum Cleaning)  
  This cleaning followed exactly the starter code
  + Dropping Records
    - Dropping records with NaN (none were found)
    - After dropping duplicates (none were found)
  + Dropping Variables that did not contribute to the variance
    - EIN
    - Name
    - STATUS
    - SPECIAL\_CONSIDERATIONS
  + Binning
    - APPLICATION\_TYPE to 9 bins
    - CLASSIFICATION to 6 bins
* Data Set Two (Maximum Cleaning)  
  This cleaning started with Data Set One and then added these steps
  + Dropping Variables that did not contribute to the variance
    - STATUS
    - SPECIAL\_CONSIDERATIONS
  + Binning
    - APPLICATION\_TYPE to 6 bins
    - CLASSIFICATION to 6 bins
    - USE\_CASE reduced to 3 bins
    - ORGANIZATION reduced to 3 bins
  + ASK\_AMT
    - Dropped records with ASK\_AMT greater than $1 billion
    - Dropped records with ASK\_AMT more than 50 times the (INCOME\_AMT)
    - Dropped records with ASK\_AMT with INCOME\_AMT = 0 and ASK\_AMT greater $100,000
* Half Data Set (Maximum Cleaning)  
  I took a 50% random sample of Data Set Two. I erroneously split the data set in half early to correct overfit (I know this is the wrong cure for overfit); however, using this data set, I got a model with 75% accuracy (but it was a random chance), that could not be reliably replicated. Therefore, I wanted to do a more robust search.

## ASK\_AMT Scaling

Both data sets still presented ASK\_AMT with a huge range that needed to be managed. I attempted two methods to wrangle this data

* Standardization (Z-Score)
  + After standardization, the Z-Score ranged from 0 to 45!
  + Dropped records with ASK\_AMT Z-Scores greater than 3 (112 records)
* Logarithmic transformation
  + I believe this is the best approach for this data; see **Error! Reference source not found.**. Nevertheless, I did the analysis with both transformation methods.

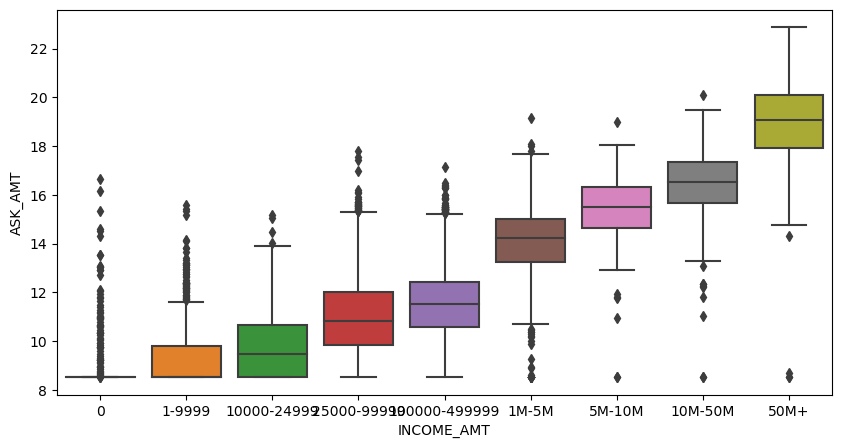


Figure : ASK\_AMT Log Scale

## Search Methods and Number of Layers

I used three types of search methods to fit the model.

* Expert Opinion  
  After considering the data set I tried various combinations of layers, neurons per layer, and activation functions that I thought would work for this data.
  + Number of layers from 2 to 3
  + Number of neurons per layer
    - Layer 1: 1x, 2x, and 3x the number of inputs
    - Layer 2: 1/2, 1/4, and 1/8th of layer 1
    - Layer 3: 2, 4, 8, 16
  + Activation functions
    - Output layer was Sigmoid
    - Interlays we mostly relu, but with more failure, I tried Sigmoid also
* Keras\_tuner
  + Using Google Colab – The problem with Colab is that my session would time out before the model would evaluate. The best case when the various attempts ended were all between 71% and 73.5%
    - Tuner = Hyperband
    - Tuner = RandomSearch
  + Using my PC, I could run longer (more than 8 hours). Unfortunately, these attempts also always crashed out. The Keras\_tuner writes a file to disk with every pass. Eventually, I would get a write error, and the process would stop.
    - Tuner = Hyperband
    - Tuner = RandomSearch
* Exhaustive
  + I built a nested for-loop model where I could control the number of layers and neurons as well as the activation functions. I tried 4 inner layers with the following numbers of neurons (124, 62, 31), (20, 10, 40), (0, 5, 15), and (0, 6). Each could have an activation function of relu or Sigmoid. This resulted in a total of 864 combinations. I ran them all.

## Activation Functions

* Output layer - In nearly all cases, the output function was sigmoid. However, I also tried tanh, relu, and softmax as output activation functions.
* Dense layers – To begin with, I focused on relu. However, all the automated testing (Keras and exhaustive use both relu and sigmoid activation functions.

# Results

## Full Data Set

I recorded 1,118 unique model fittings with the full data set (not the 50% sample). Of those 1,030 had accuracy scores greater than 0.7, Figure 2 is a histogram of the results.

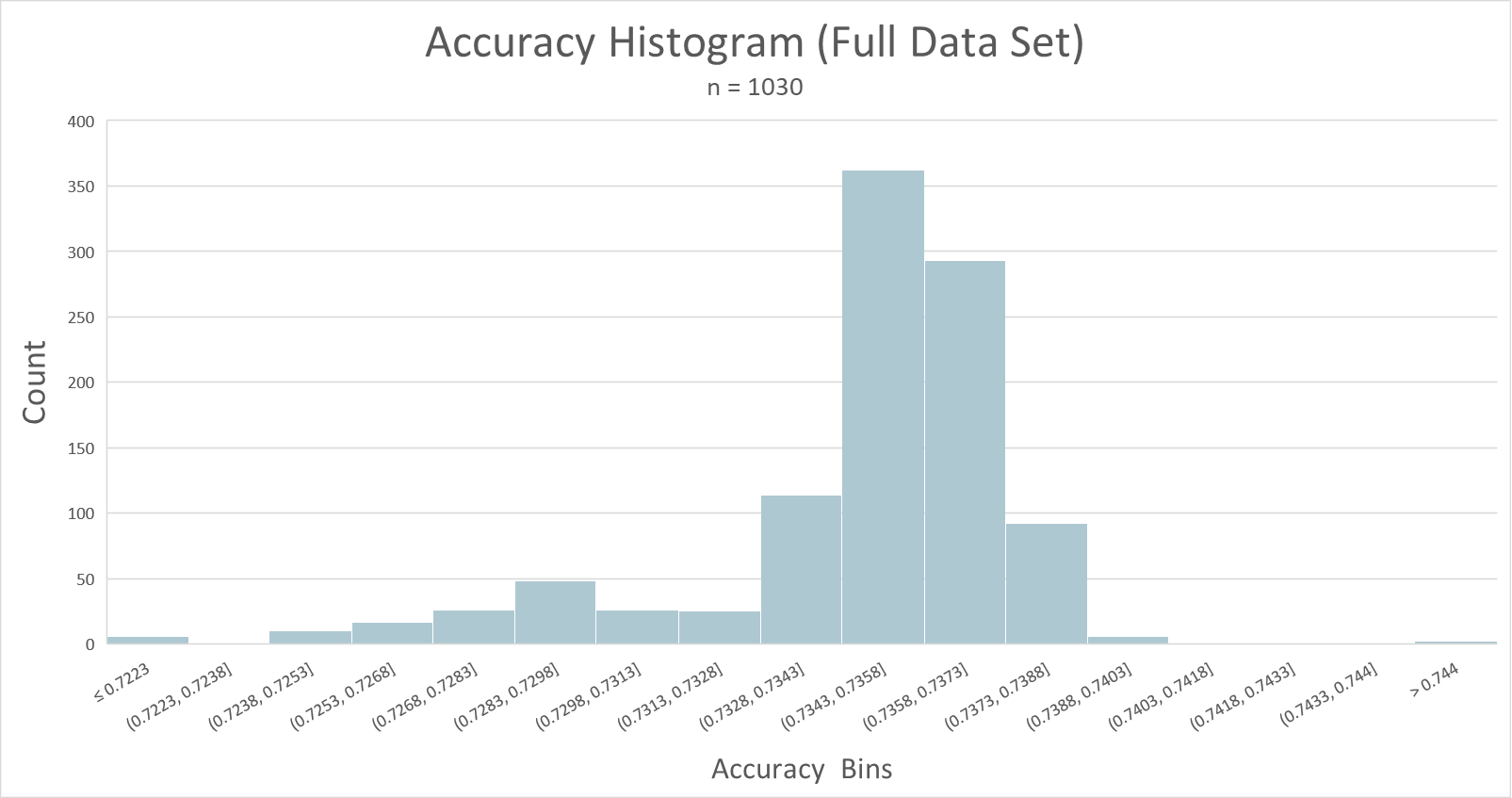


Figure : Results for Full Data Sets

The limited range of results is surprising, given the range of models that were evaluated. Table 2 identifies the minimum and max number of layers tested, while Table 3 identifies the frequency of activation functions tested.

Table : Layers (Full Data Set)



Table : Layers (Full Data Set)



## Typical Results

Chart, line chart

Description automatically generated

Figure : Typical Accuracy Plot

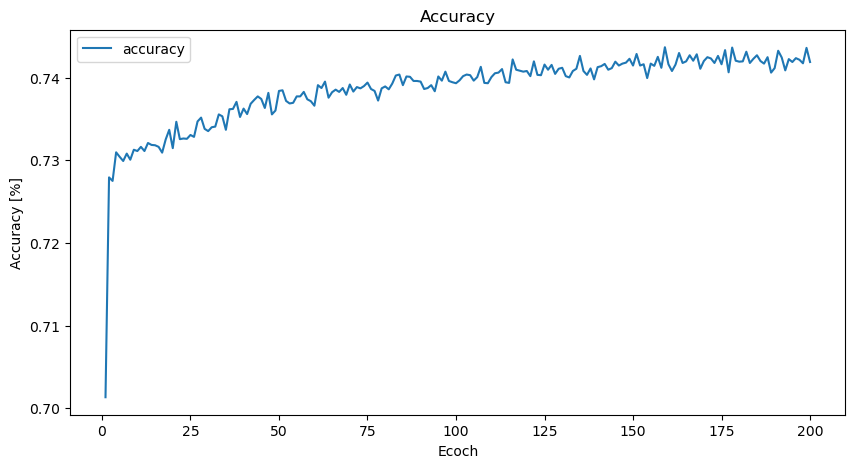


Figure : Typical Accuracy Plot

### Program files

These Jupyter notebook files are typical of the code that was created and used for this analysis. I say typical because there were many variations used and discarded because they did not make a substantial improvement over other models.

* JSP DeepLearningChallenge\_Min\_Clean.ipynb (Single pass)
* JSP DeepLearningChallenge\_Final\_Solution.ipynb (Nested for loop model)
* JSP\_21\_autotune 003.ipynb (Keras\_Tune for the PC)
* JSP\_21\_colab\_autotune 002.ipynb (Keras\_Tune for Google Colab)

### Model Results

These build/tested with the following Jupyter notebooks and save in these models.

* 2023-03-05-173048-7274-model.h5
* 2023-03-05-173048-7274-weights.hdf5

## Half Data Set

I recorded 695 unique model fittings with the half data set. Figure 3 is a histogram of the results. The limited range of results is the same as the with the full data set. Table 4 identifies the minimum and max number of layers tested, while Table 5 identifies the frequency of activation functions tested. The difference between the full data set and the falf data set is most likely due to the number of epochs. The full data set was mostly run with epoch = 50 while the half data set was run with epoch = 20.

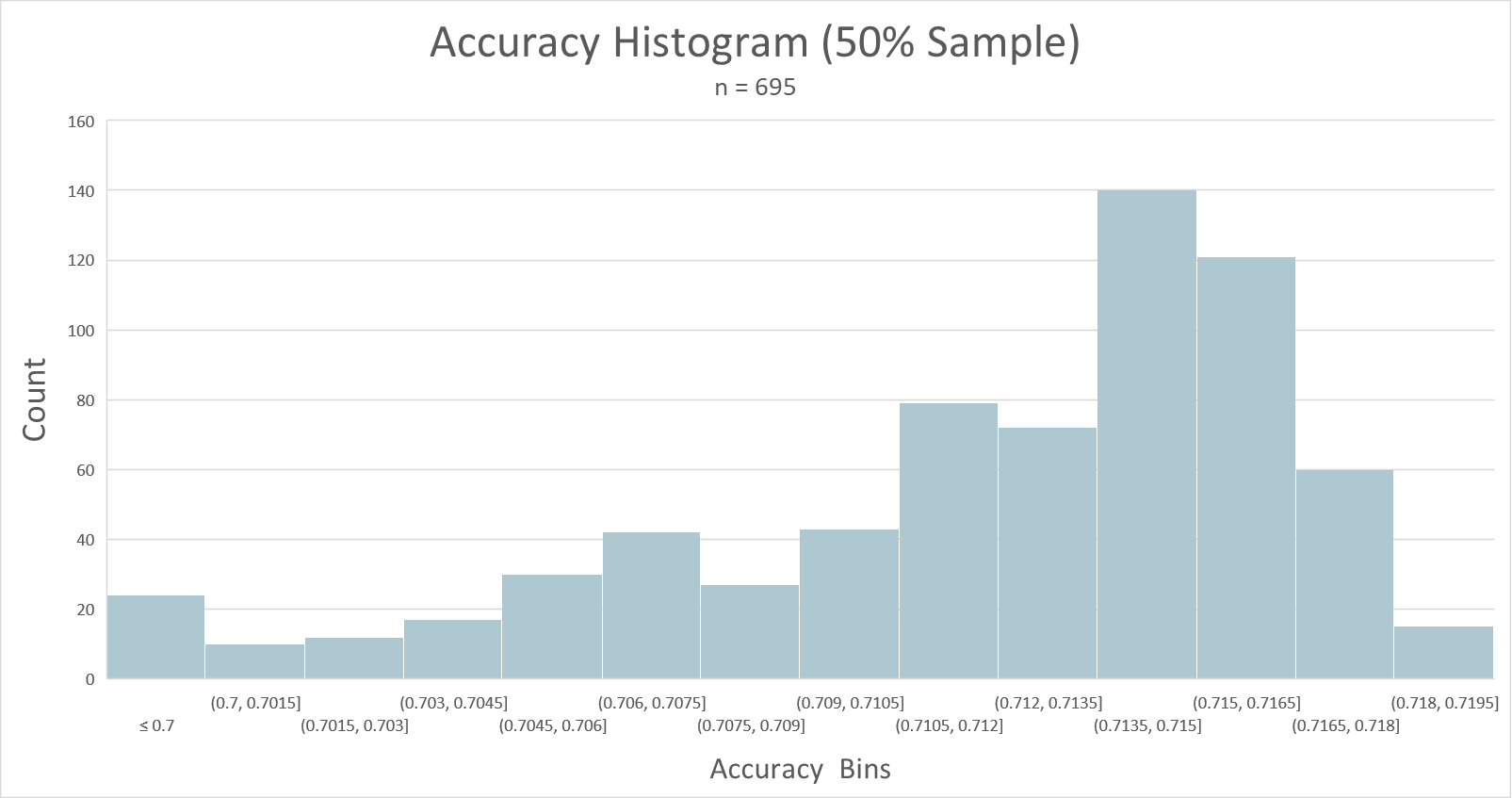


Figure : Results for Half Data Sets

Table : Layers (Half Data Set)



Table : Layers (Half Data Set)



## Random Forest

In addition to deep learning neural networks, I attempted a RandomForce model. The results were no better than the neural network solution.

True positives (TP): 3588

True negatives (TN): 2563

False positives (FP): 1474

False negatives (FN): 950

precision = 0.70881

accuracy = 0.71731

sensitivity = 0.79065

F1 = 0.7475

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | recall | f1-score | support |
| 0 | 0.73 | 0.63 | 0.68 | 4037 |
| 1 | 0.71 | 0.79 | 0.75 | 4538 |
| Accuracy |  |  | 0.72 | 8575 |
| macro avg | 0.72 | 0.71 | 0.71 | 8575 |
| weighted avg | 0.72 | 0.72 | 0.72 | 8575 |