Jeff Pinegar
jeffpinegar1@gmail.com
717-982-0516

Assignment: Deep learning Challenge
Due Date: March 6, 2023

Deep Learning Trials

Overview

Investigation of

The non-profit foundation Alphabet Soup wants a tool to help 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 achieve 75%. My best effort was 74.7%. The details for this model are covered in "Model Results" below. 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

Figures

Tables

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 summarizes those attempts.

Table 1: Attempted Models

<table>
<tr><th>Data Set Preparations</th><th>ASK_AMT Scaling</th><th>Search Method</th><th>Number of layers</th><th>Activation function</th></tr>
<tr><td rowspan="6">Data Set One
(Minimum Cleaning)</td><td>Z-Score</td><td rowspan="2">Expert Opinion</td><td rowspan="4">1 - 5</td><td rowspan="6">Relu
Sigmoid</td></tr>
<tr><td>Log</td></tr>
<tr><td>Z-Score</td><td rowspan="2">keras_tuner</td></tr>
<tr><td>Log</td></tr>
<tr><td>Z-Score</td><td rowspan="2">Exhaustive</td><td rowspan="2">1 - 4</td></tr>
<tr><td>Log</td></tr>
<tr><td rowspan="6">Data Set Two
(Maximum Cleaning)</td><td>Z-Score</td><td rowspan="2">Expert Opinion</td><td rowspan="4">1 - 5</td><td rowspan="6">Relu
Sigmoid</td></tr>
<tr><td>Log</td></tr>
<tr><td>Z-Score</td><td rowspan="2">keras_tuner</td></tr>
<tr><td>Log</td></tr>
<tr><td>Z-Score</td><td rowspan="2">Exhaustive</td><td rowspan="2">1 - 4</td></tr>
<tr><td>Log</td></tr>
<tr><td>Half Data Set
(Maximum Cleaning</td><td>Log</td><td>Exhaustive</td><td>1 - 4</td><td>Relu
Sigmoid</td></tr>
</table>

Data Set Preparations

Features and targets

For this opportunity, the data set's target (dependent) variable is IS_SUCCESSFUL. The other variables in the data set are potential independent variables or features in our model. Some of these variables will be dropped through the cleaning and transformation process, while others will be transformed.

Cleaning and transformation

This describes the cleaning 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. Unfortunately, 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. I could not be reliably replicated this result.

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 Figure 1. Nevertheless, I continued the analysis with both transformation methods.

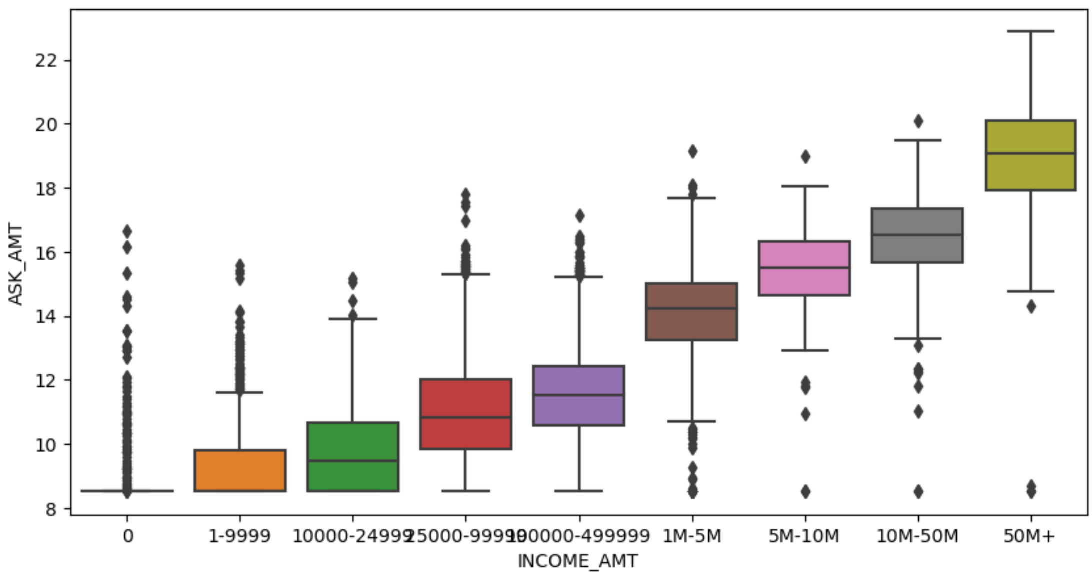

Figure 1: ASK_AMT Log Scale

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 were evaluated
    - Layer 2: 1/2, 1/4, and 1/8th of layer 1
    - Layer 3: 2, 4, 8, 16
  - Activation functions
    - Output layer was Sigmoid, since we are attempting to classify two states.
    - 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 fully evaluates. The best results were between 71% and 73.5% when the various attempts ended. I used two different tuners for this search:
    - Tuner = Hyperband
    - Tuner = RandomSearch
  - I could run longer (more than 8 hours) using my PC. Unfortunately, these attempts also eventually crashed. The Keras_tuner writes a file to disk with every pass. Eventually, I would get a write error, and the process would stop. The best results were between 71% and 73.5% when the various attempts ended. I used two different tuners for this search:
    - Tuner = Hyperband
    - Tuner = RandomSearch
- Exhaustive
  - To avoid the write file problem, I was experiencing with Keras_tuner, 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. Sigmoid and Tanh were selected because they could make the final classification for IS_SUCCESSFUL or not.
- Dense layers – To begin with, I focused on Relu. However, all the automated testing (Keras and exhaustive) use Relu and sigmoid activation functions in the dense layers.

Results

Full Data Set

I recorded 1,118 unique model fittings with the full data set (not the 50% sample). Of those, 1,030 trials had accuracy scores greater than 0.7. Figure 2 is a histogram of the results (with a bucket for less than 72.23%).

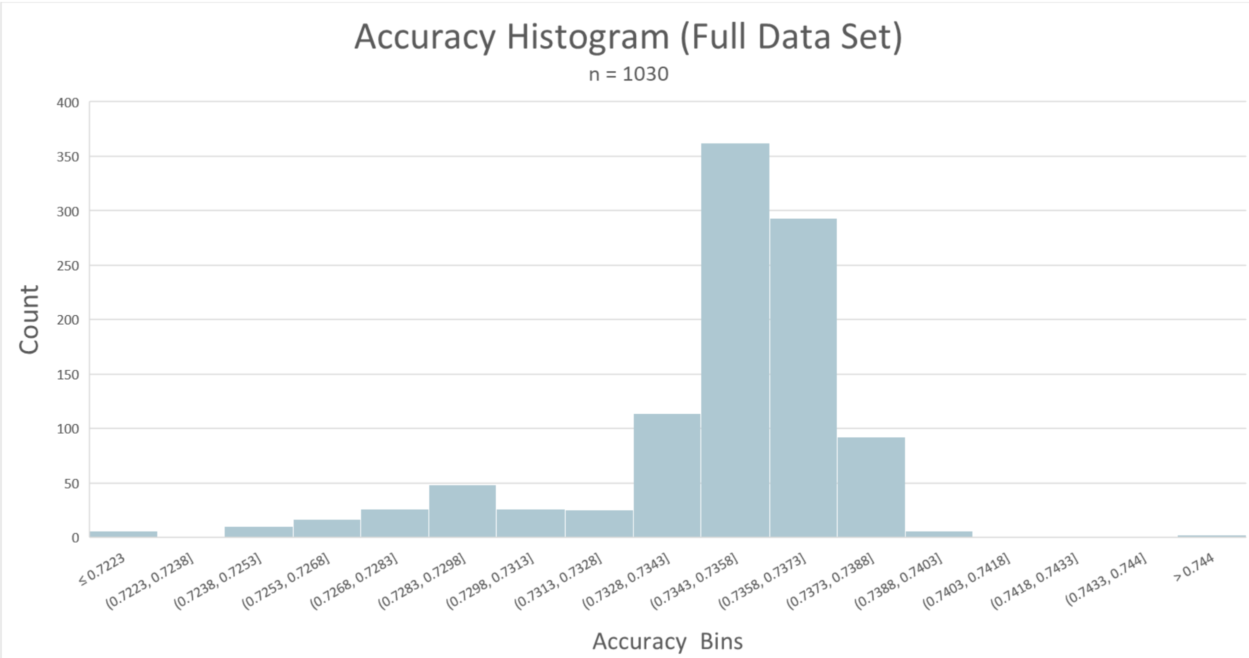

Figure 2: Results for Full Data Sets

The limited range of results is surprising, given the range of models that were evaluated. Table 2 identifies the minimum and maximum number of layers tested, while Table 3 identifies the frequency of activation functions tested. The spreadsheet Report figure.xls has the details for the 1,118 runs.

Table 2: Layers (Full Data Set)

No. Neurons	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Min	4	4	0	0	0	0
Max	150	100	100	20	20	20
Count	1030	1030	684	533	8	5

Table 3: Layers (Full Data Set)

	relu	sigmoid
layer 1	607	421
layer 2	544	482
layer 3	342	339
layer 4	273	256
layer 5	8	0
layer 6	5	0

Typical Results

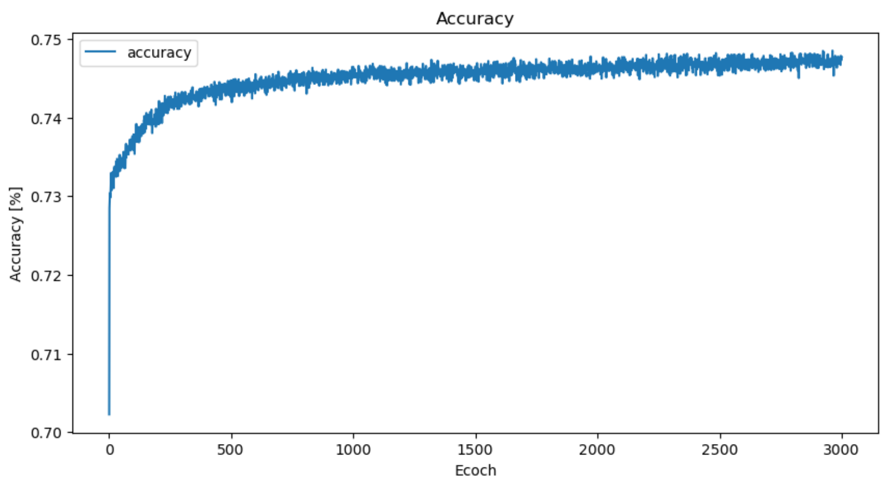

Figure 3: Typical Accuracy Plot

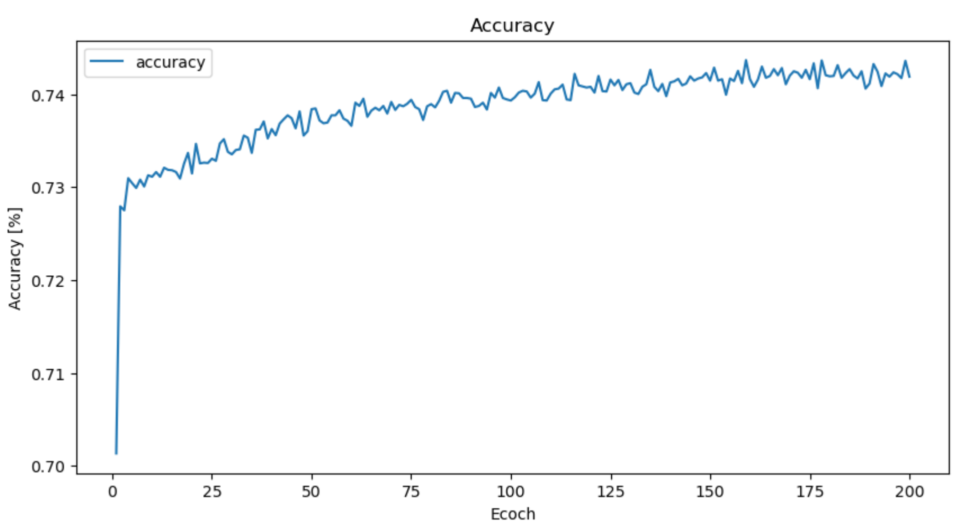

Figure 4: Typical Accuracy Plot

Program files

These Jupyter notebook files are typical of the code created and used for this analysis. I say typically because many variations were used and discarded because they did not make a substantial improvement over other models. See Figure 5: Accuracy Plot for 74.7% Model and Figure 6: Loss Plot for 74.7% Model

- 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

The single-pass jupyter notebook was used to build these models after exploratory research was completed using the other program to select the model. I then ran that configuration with 3,000 epochs. The result was a model with 74.7% accuracy.

- 2023-03-05-221906-7234-Min clean 747 model.h5
- 2023-03-05-221906-7234-Min Clean 747 weights.hdf5

My best (most accurate) model had the following parameters

inputs = 40	Number of independent variables in the data frame
lay_1_n = 102	Number of neurons in the first layer
lay_2_n = 10	Number of neurons in the second layer
lay_3_n = 10	Number of neurons in the second layer
lay_4_n = 2	Number of neurons in the second layer
act1 = "relu"	Layer 1 activation function
act2 = "sigmoid"	Layer 2 activation function
act3 = "relu"	Layer 3 activation function
act4 = "relu"	Layer 4 activation function
actout = "sigmoid"	Output activation function

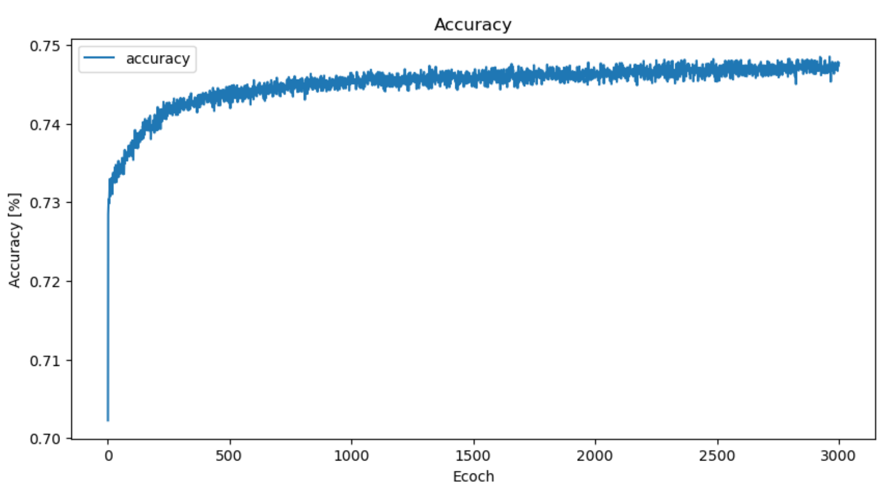

Figure 5: Accuracy Plot for 74.7% Model

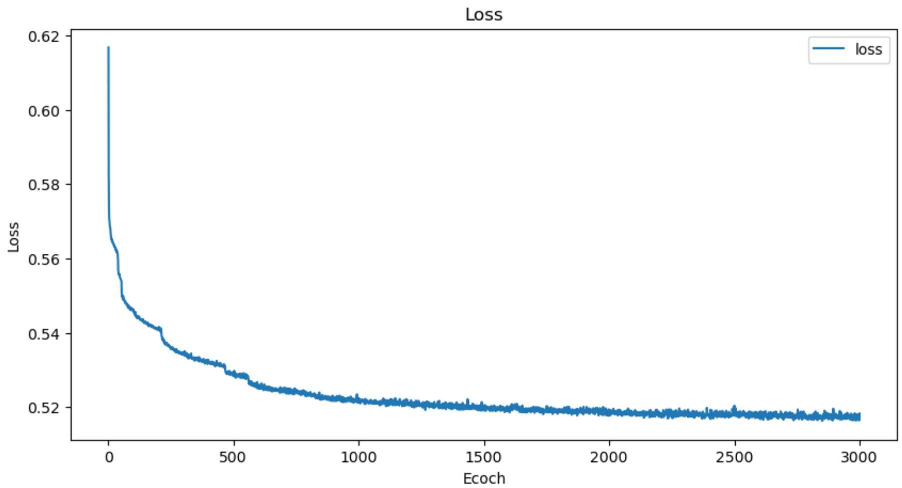

Figure 6: Loss Plot for 74.7% Model

Half Data Set

I recorded 695 unique model fittings with the half data set. Figure 7 is a histogram of the results. The limited range of effects is the same as the entire 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 and half data sets can most likely be explained by the number of epochs. The full data set was mostly run with epoch = 50, while the half data set was run with epoch = 20.

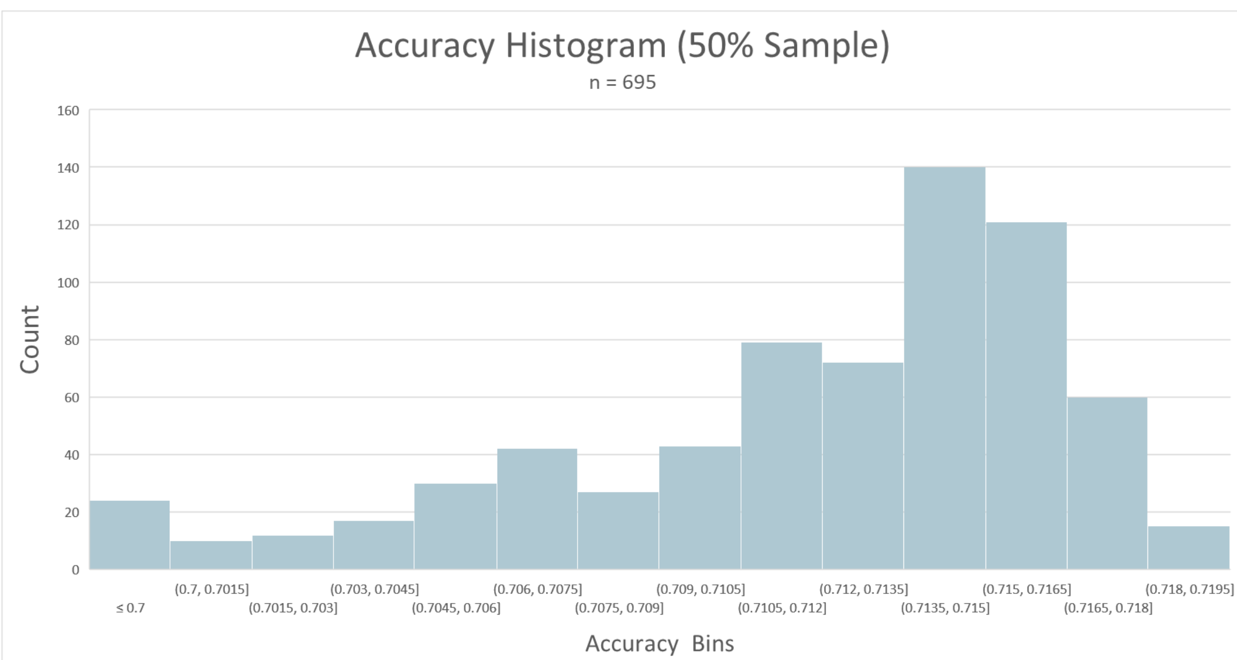

Figure 7: Results for Half Data Sets

Table 4: Layers (Half Data Set)

No. Neurons	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Min	4	4	0	0	0	0
Max	150	100	100	20	20	20
Count	1030	1030	684	533	8	5

Table 5: Layers (Half Data Set)

	relu	sigmoid
layer 1	607	421
layer 2	544	482
layer 3	342	339
layer 4	273	256
layer 5	8	0
layer 6	5	0

Random Forest

In addition to deep learning neural networks, I attempted a RandomForest model. Unfortunately, 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

Reference List of files

Project Files

Challenge 21 - Deep Learing.pdf
Original instructions for the challenge

Challenge 21 - Deep Learning HW Rubric - Charity Funding Predictor.pdf
Original project scoring rubric

JSP Challenge 21 Deep Learning Report.docx
This document

Report Figures.xlsx
This Excel file contains the record of more than 2000 model trials. In addition, it has several of the data visualization used to create this report.

Python Code File

Starter_Code.ipynb
This is the original starter code for this challenge

JSP DeepLearningChallenge_Min_Clean.ipynb
This is the best model that I was able to produce. The parameters were selected based on the exploratory work done using the other Juypter notebooks.

JSP DeepLearningChallenge_Max_Clean.ipynb
This code was built to perform model fitting using the maximum data cleaning and transformation.

JSP DeepLearningChallenge_For_Loop.ipynb
This code starts with the data clean by JSP DeepLearningChallenge_Min_Clean.ipynb then using nested for-loops test over 800 combinations for layers, neurons, and activation functions.

JSP DeepLearningChallenge_For_Loop_half.ipynb
This code starts with the data clean by JSP DeepLearningChallenge_Max_Clean.ipynb then using nested for-loops test over 800 combinations for layers, neurons and activation functions.

JSP DeepLearningChallenge_RandomForest-2.ipynb
Since Neural Networks were not yielding a model with the desired accuracy, I tried a RandomForest model. The results were not better or worse.

JSP_21_autotune 003.ipynb
Code set up to test using Keras_tuner on my PC.

JSP_21_colab_autotune 002.ipynb
Code set up to test using Keras_tuner in Google Colab.

Comma Separated Value files

clean_data_all_log_dummies.csv
Data set that was cleaned, and the ASK_AMT logarithmically transformed. The other variables were not scaled since they were all 0 or 1.

clean_data_Reduced_log_dummies.csv
Data set that was cleaned, and the ASK_AMT logarithmically transformed. The other variables were not scaled since they were all 0 or 1. This is then a 50% random sample of that data set.

clean_data_reduced_stand_dummies.csv
Data set that was cleaned. Everything was then scaled using standard (Z-score) scaling.

Models

2023-03-05-221906-7234-Min clean 747 model.h5
2023-03-05-221906-7234-Min Clean 747 weights.hdf5
This is the best-fit model from the entire data set with minimal data cleaning. It has an accuracy of 74.7%

2023-03-06-074225-7362-MaxClean-model.h5
2023-03-06-074225-7362-MaxClean-weights.hdf5
This is the best-fit model from the entire data set with minimal data cleaning. It has an accuracy of 73.6%