

Module 21 - Deep Learning Report

Overview

The following report will cover different aspects of the Module 21 Neural Networks assignment. This data will include an explanation of the purpose of the assignment, as well as the types of preprocessing that were completed, and lastly an analysis of the findings after attempting 4 variants of the neural network. The purpose of this assignment was to import a CSV named "charity_data.csv" while using Google Collab, preprocessing the data, and training a neural network to predict what makes a charity campaign a success. The CSV itself contained 34,299 rows of data with no nulls and mostly integer or float data types. Those that were not integers or floats were appropriate data types for the characteristic.

Initial Steps and Data Evaluation

After importing the CSV, preprocessing actions took place. Imports were added to the notebook to include several modules from scikit learn regarding classification and encoding modules. Tensorflow, matplotlib and pandas were imported as well amongst others. It was at this time that the data was reviewed for its information, where it could be verified that no nulls were present, and that all data types were correct. From here analysis was done on the dataframe containing the data to review the mean, median, and count to better understand whether scaling was going to be required. The data suggested that several columns would require scaling to ensure that the characteristics would be useful to the network. Value counts were pulled for every column to enable easier comprehension of areas where bucketing and label encoding might enable a reduction in the number of variables for each column.

Preprocessing

Once these areas for refinement were defined, the columns of "EIN," "NAME", "SPECIAL_CONSIDERATIONS", and "STATUS", were dropped from the dataframe as a subsequent review suggested that these would not be value added when training a neural network. Additional preprocessing saw the "APPLICATION_TYPE" column have several of its variable merged into a single label title "Other." This was done because there were large gulfs between several of the types of application, with many campaigns only having 1 or 2 entries, while others had 27,037 lines of data. As a result, all application types that had value counts under 600 were merged into the aforementioned label "Other." This same process was completed for the "CLASSIFICATION" column, with the exception that the "threshold" was value counts under 500.

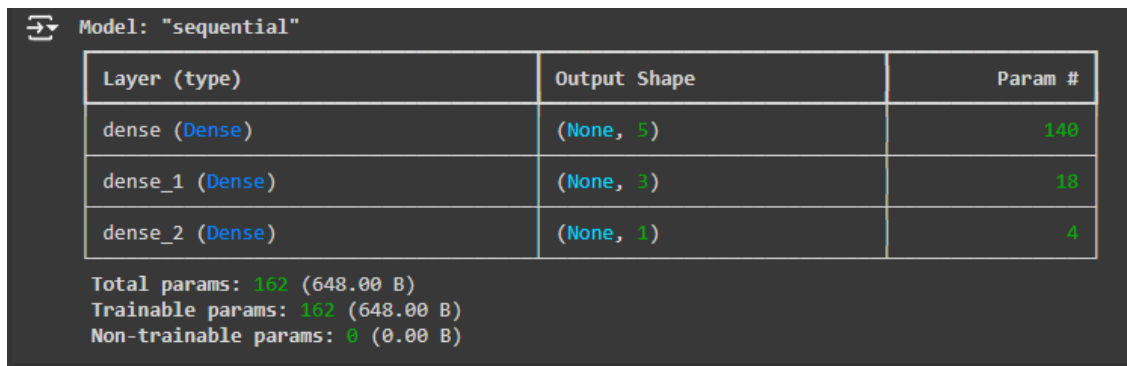
With the data consolidated a little bit more, the dataframe had the columns broken down into separate ones for each value that was present. These were in turn using OneHot Encoding to apply binary classifications stating whether the relevant entry was True (1) or False (0). Any column that showed True meant that that characteristic was present. This reduced complexity and turned a column that had been strings / booleans into binary making it more useful for the network. After applying the OneHot Encoding, the "IS_SUCCESSFUL" column was dropped, as that would serve as the objective for the model to start predicting. Once Standard Scaling was applied, the model was compiled, and the training began.

Four separate models were generated, each with their own set of parameters.

Neural Network Model 1:

- Preprocessing: Columns "EIN," "NAME", "SPECIAL_CONSIDERATIONS", and "STATUS" were dropped
- Number of Layers: 2 hidden layers
- Node Counts:
 - Layer 1: 5 nodes
 - Layer 2: 3 nodes
- Activation: Rectified Linear Unit (ReLU)
- Number of Epochs: 500
- Results: Overall, this model only had 46% accuracy, with around 77% loss.

Fig 1 – Model 1



The image shows a screenshot of a Keras model summary for a sequential model. The model is named "sequential". It consists of three layers: a dense layer with 5 nodes, a dense layer with 3 nodes, and a dense layer with 1 node. The total number of parameters is 162, with 162 trainable parameters and 0 non-trainable parameters. The output shape for each layer is (None, 5), (None, 3), and (None, 1) respectively.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 5)	140
dense_1 (Dense)	(None, 3)	18
dense_2 (Dense)	(None, 1)	4

Total params: 162 (648.00 B)
Trainable params: 162 (648.00 B)
Non-trainable params: 0 (0.00 B)

Neural Network Model 2:

- Preprocessing: Column "ORGANIZATION" was dropped
- Number of Layers: A third hidden layer was added
- Node Counts:
 - Layer 1: 9 nodes
 - Layer 2: 7 nodes
 - Layer 3: 5 nodes.
- Activation: ReLU
- Number of Epochs: 500
- Results: Overall, this model was significantly better, accuracy got up to 62%, with around 98% loss.

Fig 2 – Model 2

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 9)	252
dense_4 (Dense)	(None, 7)	70
dense_5 (Dense)	(None, 5)	40
dense_6 (Dense)	(None, 1)	6

Total params: 368 (1.44 KB)
Trainable params: 368 (1.44 KB)
Non-trainable params: 0 (0.00 B)

Neural Network Model 3:

- Preprocessing: None
- Number of Layers: A fourth hidden layer was added
- Node Counts:
 - Layer 1: 9 nodes
 - Layer 2: 7 nodes
 - Layer 3: 5 nodes
 - Layer 4: 3 nodes.
- Activation: Hyperbolic Tangent Function (TanH)
- Number of Epochs: 500
- Results: This model performed worse than model 2, but better than model 1, coming in at 51% accuracy, with a 69% loss.

Fig 3 – Model 3

Model: "sequential_2"

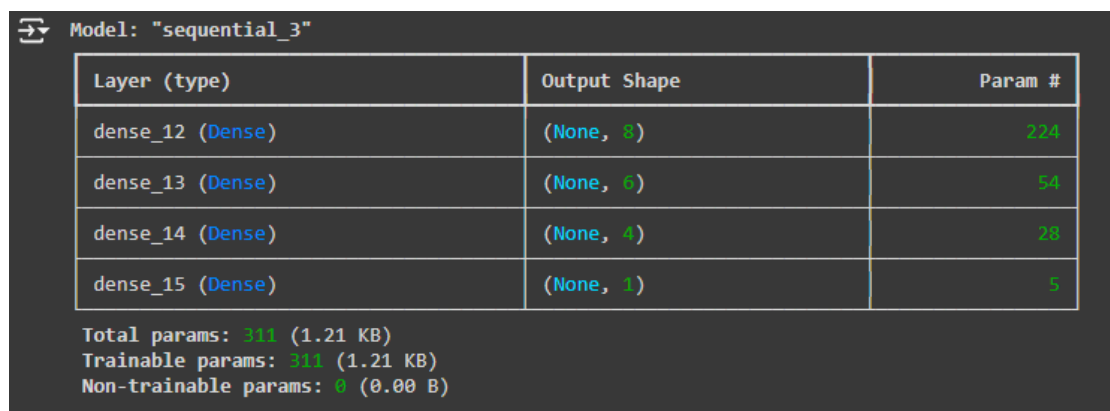
Layer (type)	Output Shape	Param #
dense_7 (Dense)	(None, 9)	252
dense_8 (Dense)	(None, 7)	70
dense_9 (Dense)	(None, 5)	40
dense_10 (Dense)	(None, 3)	18
dense_11 (Dense)	(None, 1)	4

Total params: 384 (1.50 KB)
Trainable params: 384 (1.50 KB)
Non-trainable params: 0 (0.00 B)

Neural Network Model 4:

- Preprocessing: None
- Number of Layers: The fourth hidden layer was dropped
- Node Counts:
 - Layer 1: 8 nodes
 - Layer 2: 6 nodes
 - Layer 3: 4 nodes
- Activation: Sigmoid
- Number of Epochs: 500
- Results: Overall, this was the second-best model, with 53% accuracy and 69% loss.

Fig 4 – Model 4



The image shows a screenshot of a Keras model summary for a model named "sequential_3". The summary is displayed in a dark-themed interface. It includes a table with three columns: "Layer (type)", "Output Shape", and "Param #". The table lists four dense layers: dense_12 (Dense) with output shape (None, 8) and 224 parameters; dense_13 (Dense) with output shape (None, 6) and 54 parameters; dense_14 (Dense) with output shape (None, 4) and 28 parameters; and dense_15 (Dense) with output shape (None, 1) and 5 parameters. Below the table, the summary provides totals: Total params: 311 (1.21 KB), Trainable params: 311 (1.21 KB), and Non-trainable params: 0 (0.00 B).

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 8)	224
dense_13 (Dense)	(None, 6)	54
dense_14 (Dense)	(None, 4)	28
dense_15 (Dense)	(None, 1)	5

Total params: 311 (1.21 KB)
Trainable params: 311 (1.21 KB)
Non-trainable params: 0 (0.00 B)

Evaluation:

As all four models had now been run, it was relevant to try and understand what happened with this dataset. Per the objectives laid out in the instructions, the goal was to have the model come back with 75% accuracy. The most successful of my models was Model 2, which came in at 62% accuracy. While there are several factors that could be impacting these results, I believe that because each of the columns have significant outliers was drastically impairing the ability of the model to more accurately predict what the outcome would be. It is possible that if the "APPLICATION_TYPE" and "CLASSIFICATION" columns were to be dropped then there might be more relevant data. But in terms of doing this in support of a fictional company, removing the aforementioned columns could result in losing relevant context.

Final Thoughts

Overall, this has been an interesting experiment to work on. Getting a chance to interact with machine learning has helped me have a better appreciation for the importance of data cleaning when dealing with this material. There is some additional fine tuning that could be done to try to get these models to be more useful, but I felt