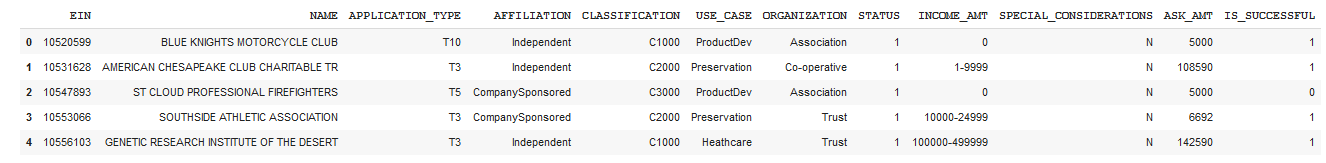
**Overview**

* ***The purpose of this repo was to create a model that would determine whether or not applications for funding would be successful or not***
* The data used for this repo was from Alphabet Soup
* It was complicated data with lots of different fields that were not organized neatly
  + That being said, most machine learning models other than neural networks such as linear regression, logistic regression, k-nearest neighbors, and many others would not have been effective predicting this model.
  + My neural network did not prove to be a very effective model either but it performed better than many other model types would have.

**Results**

**Breaking down the CSV Columns**

**Target Column:**

* IS\_SUCCESSFUL

**Feature Columns**

* APPLICATION\_TYPE, AFFFLIATION, CLASSIFICATION, USE\_CASE, ORGANIZATION, STATUS, INCOME\_AMT, SPECIAL\_CONSIDERATIONS, ASK\_AMT

**Neither** (columns that were removed from dataset before creating model)

* EIN and NAME

**Attempts and Results**

**Initial Model**

Application Bins Classification Bins

A picture containing table

Description automatically generatedText, table

Description automatically generated

Number of Hidden Layers 2

Number of Epochs 100

**First Layer**

* Number of Nodes 80
* Activation Function Relu

**Second Layer**

* Number of Nodes 30
* Activation Function Relu

**Output Layer**

* Activation Function Sigmoid

**Results**

Loss: 0.561 Accuracy: 0.7267



This model is not great, so I tried to change the model’s parameters to see if I could improve the accuracy.

**Improvement Attempt 1**

*Add another hidden layer*

Application Bins (same) Classification Bins

A picture containing table

Description automatically generatedText, table

Description automatically generated

Number of Hidden Layers 3

Number of Epochs 100

**First Layer**

* Number of Nodes 80
* Activation Function Relu

**Second Layer**

* Number of Nodes 30
* Activation Function Relu

**Third Layer**

* Number of Nodes 10
* Activation Function Relu

**Output Layer**

* Activation Function Sigmoid

**Results**

Loss: 0.558 Accuracy: 0.725



Adding another hidden layer to the neural network had such a small effect on the model's results (decrease in accuracy of 0.001) that it is insignificant. That being said, having another hidden layer takes more computational power, so I will continue to use 2 hidden layers not 3

**Improvement Attempt 2**

*Change the activation functions of one of the hidden layers from relu to tanh*

Application Bins Classification Bins

A picture containing table

Description automatically generatedText, table

Description automatically generated

Number of Hidden Layers 2

Number of Epochs 100

**First Layer**

* Number of Nodes 80
* Activation Function Relu

**Second Layer**

* Number of Nodes 30
* Activation Function Tanh

**Output Layer**

* Activation Function Sigmoid

**Results**

Loss: 0.554 Accuracy: 0.727



Similar to the first attempt, changing the activation function from relu to tanh had such a small effect on the model that it is practically insignificant. However, even though the change was extremely small the model did perform better with tanh than relu so I am keep tanh as the activation function in my next attempt.

**Improvement Attempt 3**

*Decrease the number of the nodes in both hidden layers*

Application Bins Classification Bins

A picture containing table

Description automatically generatedText, table

Description automatically generated

Number of Hidden Layers 2

Number of Epochs 100

**First Layer**

* Number of Nodes 20
* Activation Function Relu

**Second Layer**

* Number of Nodes 10
* Activation Function Tanh

**Output Layer**

* Activation Function Sigmoid

**Results**

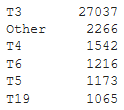
Loss: 0.553 Accuracy: 0.726



Similar to the first 2 attempts, changing the number of nodes in the hidden layers had such a small effect on the model that it is practically insignificant. The accuracy of this model is 0.001 worse than improvement attempt 2

**Improvement Attempt 4**

*Decrease the number of bins*

Text

Description automatically generated with low confidenceApplication Bins Classification Bins

Number of Hidden Layers 2

Number of Epochs 100

**First Layer**

* Number of Nodes 20
* Activation Function Relu

**Second Layer**

* Number of Nodes 10
* Activation Function Tanh

**Output Layer**

* Activation Function Sigmoid

**Results**

Loss: 0.563 Accuracy: 0.7259



Similar to the first 23 attempts, changing the number of bins had such a small effect on the model that it is practically insignificant. The accuracy of this model is 0.001 worse than improvement attempt 2

**Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| Overall Results | | | |
| **Model** | **Change** | **Loss** | **Accuracy** |
| Initial Model | None | 56.1% | 72.67% |
| Attempt 1 | Add another hidden layer | 55.8% | 72.55% |
| Attempt 2 | Change the activation functions of one layers from relu to tanh | 55.4% | 72.71% |
| Attempt 3 | Decrease the number of the nodes in both hidden layers | 55.3% | 72.62% |
| Attempt 4 | Decrease the number of bins | 56.3% | 72.59% |

Overall, I am very surprised and almost disappointed with how my neural network performed. Not that the models performed awfully (I mean they did not perform great either), but more that fact that after 4 attempts to improve the model not only did the model not improve, but it really did not change at all either. All 5 of the neural networks that I ran had an accuracy within 0.2% of one another and a loss with 1%. Yes, small percentages like this do matter when working with large datasets, but this was a relatively small dataset so the alterations that I made in an attempt to improve the model were clearly not significant enough.

If I had to recommend a model (other than a neural network) to use to predict this dataset. I would recommend a Random Forests Model because this is a classification problem and Random Forests Model is one of the best classification models