

GREY HARDY  
11/15/2021

# Deep Learning Project: Charity Funding Predictor

Deep learning and neural networks were used to determine if applicants would be successfully funded by Alphabet Soup, whom previously funded over 34,000 organizations.

## Data Processing

The dataset removed any irrelevant information; therefore, EIN and NAME were dropped from the model. The remaining columns were considered features for the model. Although NAME was added back in the second test. CLASSIFICATION and APPLICATION\_TYPE was replaced with 'Other' due to high fluctuation. The data was split into training and testing sets of data. The target variable for the model is "IS\_SUCCESSFUL" and is verified by the value, 1 was considered yes and 0 was no. APPLICATION data was analyzed, and CLASSIFICATION's value was used for binning. Each unique value used several data point as a cutoff point to bin "rare" categorical variables together in a new value, 'Other'. Afterwards checked to see if binning was successful. Categorical variables were encoded by 'pd.get\_dummies()'.

## Compiling, Training, and Evaluation the Model

Neural Network was applied on each model multiple layers, three in total. The number of features dictated the number of hidden nodes.

```
[ ] # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
    number_input_features = len( X_train_scaled[0])
    hidden_nodes_layer1=7
    hidden_nodes_layer2=14
    hidden_nodes_layer3=21
    nn = tf.keras.models.Sequential()

    nn = tf.keras.models.Sequential()

    # First hidden layer
    nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))

    # Second hidden layer
    nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))

    # Output layer
    nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

    # Check the structure of the model
    nn.summary()
```

A three-layer training model generated 477 parameters. The first attempt came close at 72% which was under the desired 75%.

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	350
dense_1 (Dense)	(None, 14)	112
dense_2 (Dense)	(None, 1)	15

=====  
Total params: 477  
Trainable params: 477  
Non-trainable params: 0

```
[ ] # Evaluate the model using the test data
    model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
    print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

268/268 - 0s - loss: 0.5519 - accuracy: 0.7278 - 283ms/epoch - 1ms/step  
Loss: 0.5519309639930725, Accuracy: 0.7278134226799011

## Optimization

The second attempt added 'NAME' back into the dataset, this time I achieved 79% which was 4% over target. A total of 3,298 params.

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	3171
dense_1 (Dense)	(None, 14)	112
dense_2 (Dense)	(None, 1)	15

=====  
Total params: 3,298  
Trainable params: 3,298  
Non-trainable params: 0

Deep learning models should have multiple layers, since it is machined based it teaches a computer to filter inputs through the layers to learn how to predict and classify information.



```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - loss: 0.4647 - accuracy: 0.7869 - 229ms/epoch - 856us/step
Loss: 0.4646620452404022, Accuracy: 0.7869387865066528
```