

Deep Learning- Charity Funding Predictor

Overview:

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

Results:

Data Processing

Any irrelevant information was removed from the dataset. After dropping EIN and NAME, the remaining columns were considered the 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" which 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 variable together in a new value, 'Other'. Afterwards, binning was checked for success. In addition, the categorical variables were encoded by `pd.get_dummies()`.

Compiling, Training, and Evaluating the Model

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

```
1 # Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
2 # YOUR CODE GOES HERE
3 number_input_features = len( X_train_scaled[0])
4 hidden_nodes_layer1=7
5 hidden_nodes_layer2=14
6 hidden_nodes_layer3=21
7 nn = tf.keras.models.Sequential()
8
9 nn = tf.keras.models.Sequential()
10
11 # First hidden layer
12 # YOUR CODE GOES HERE
13 nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation='relu'))
14
15 # Second hidden layer
16 # YOUR CODE GOES HERE
17 nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation='relu'))
18
19 # Output layer
20 # YOUR CODE GOES HERE
21 nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
22
23 # Check the structure of the model
24 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_3"

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 7)	350
dense_4 (Dense)	(None, 14)	112
dense_5 (Dense)	(None, 1)	15
Total params: 477		
Trainable params: 477		
Non-trainable params: 0		

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

268/268 - 0s - loss: 0.5545 - accuracy: 0.7270
Loss: 0.5545142292976379, Accuracy: 0.7269970774650574

Optimization

The second attempt added 'NAME' back into the dataset and achieved 79%, which was over 4% 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		

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

268/268 - 0s - loss: 0.4711 - accuracy: 0.7865
Loss: 0.4710586667060852, Accuracy: 0.7864723205566406

Summary:

Multiple layers should be used for deep learning models since it learns how to predict and classify information based on computer filtering inputs through layers.