Neural Network Charity Analysis

By Veronica Ostapowich

A **Neural Network Binary Classifier** was created to predict if the nonprofit foundation Alphabet Soup charities will be successful if they receive funding and optimize this model to achieve higher than 75% accuracy.

Overview of the analysis

This project includes Jupyter Notebook files to build, train, test, and optimize a deep neural network that models charity success from nine features in a loan application data set. We used the TensorFlow Keras Sequential model with Dense hidden layers and a binary classification output layer and optimized this model by varying the following parameters:

- Training duration in epochs
- Hidden layer architecture
- Hidden layer activation functions
- Categorical variable bucketing
- Number of input features
- Batch size
- Learning rate

Resources

Data Source:	https://static.bc-edx.com/data/dl-1-2/m21/lms/starter/charity_data.csv
Software:	Google Colab, Jupyter Notebook, pandas, Python, scikit-learn, TensorFlow

Below is a sample of our data:

	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CASE	ORGANIZATION	STATUS	INCOME_AMT	SPECIAL_CONSIDERATIONS	ASK_AMT	IS_SUCCESSFUL
0	T10	Independent	C1000	ProductDev	Association	1	0	N	5000	1
1	T3	Independent	C2000	Preservation	Co-operative	1	1-9999	N	108590	1
2	T5	CompanySponsored	C3000	ProductDev	Association	1	0	N	5000	0
3	T3	CompanySponsored	C2000	Preservation	Trust	1	10000-24999	N	6692	1
4	T3	Independent	C1000	Heathcare	Trust	1	100000-499999	N	142590	1
...	...	...	...	...	...	...	...	...	...	...
34294	T4	Independent	C1000	ProductDev	Association	1	0	N	5000	0
34295	T4	CompanySponsored	C3000	ProductDev	Association	1	0	N	5000	0
34296	T3	CompanySponsored	C2000	Preservation	Association	1	0	N	5000	0
34297	T5	Independent	C3000	ProductDev	Association	1	0	N	5000	1
34298	T3	Independent	C1000	Preservation	Co-operative	1	1M-5M	N	36500179	0

Data Preprocessing

STEP 1. Preprocess our data set ***charity_data.csv*** by reading our data and noting the following information:

Target Variable:	IS_SUCCESSFUL	
Identification Variables to be removed:	EIN, NAME	
Feature Variables:	• APPLICATION_TYPE • AFFILIATION • CLASSIFICATION • USE_CASE • ORGANIZATION	• STATUS • INCOME_AMT • SPECIAL_CONSIDERATIONS • ASK_AMT

STEP 2. Convert categorical data to numeric with ***pd.get_dummies*** after bucketing noisy features APPLICATION_TYPE, ASK_AMT, and CLASSIFICATION with many unique values.

STEP 3. We split our preprocessed data into our features and target arrays, split further into training and testing datasets, and scaled our training and testing data.

Compiling, Training, and Evaluating the Model

Using our knowledge of ***TensorFlow***, we designed a neural network, or deep learning model, to create a binary classification model that can predict if an "Alphabet Soup" funded organization would be successful based on the dataset features. We determined how many inputs there were before deciding the number of neurons and layers in the first model. We compiled, trained, and evaluated the binary classification model to calculate the model's loss and accuracy with the following parameters:

Parameter	Value	Justification
Number of Hidden Layers	Two	A Deep Neural Network is necessary for complex data. This is a good starting point with low computation time.
Architecture	hidden_nodes1 = 10 hidden_nodes2 = 20	The first layer was set at 10 (we have nine featured variables), and a second layer was added to offer a shorter computation time.
Hidden Layer Activation Function	relu	A generic choice for inexpensive training with generally good performance.
Number of Output Nodes	One	Therefore, this binary classifier model should have one output predicting if the variable IS_SUCCESSFUL is True or False.
Output Layer Activation Function	sigmoid	It provides a probability output (values between 0 and 1) for classifying our target variable.
Training Duration (epochs)	50	Initial training duration

This parameter combination yielded the model summary shown in the Base Model summary and was saved as ***AlphabetSoupCharity.ipynb***:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                510

 dense_1 (Dense)             (None, 20)                220

 dense_2 (Dense)             (None, 1)                 21

=================================================================
Total params: 751
Trainable params: 751
Non-trainable params: 0
_________________________________________________________________
```

We then compiled and trained the model using the ***binary_crossentropy*** loss function, ***adam*** optimizer, and ***accuracy*** metric to obtain the training results shown in our Base Model Training. Verifying with the testing set, we got the following results:

```
268/268 - 1s - loss: 0.5452 - accuracy: 0.7324 - 584ms/epoch - 2ms/step
Loss: 0.5451513528823853, Accuracy: 0.7323614954948425
```

Since our accuracy is not reaching the goal of achieving more than 75% accuracy, we optimized the previous model by adjusting the parameters shown above.

We saved it as ***AlphabetSoupCharity_Optimization.ipynb***, initially making the following single changes:

Parameter	Value Change To	Justification
Architecture	Adding a third hidden layer	Adding a third hidden layer to a deep learning model can provide additional capacity and flexibility, allowing it to learn more complex and abstract features from the input data.
Hidden Layer Activation Function	The second and third layers were changed to tanh	Using different activation functions on the hidden layers in a deep learning model is to introduce non-linearity into the network.
Number of input features	hidden_nodes_layer1 = 80 hidden_nodes_layer2 = 50 hidden_nodes_layer3 = 30	Increasing the input features per hidden layer allows us to capture complex relationships, learn hierarchical representations, avoid overfitting, and improve the overall performance.

The parameters changed yielded our optimized model summary shown in our Base Model Summary.

```
Model: "sequential_62"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_205 (Dense)           (None, 80)                4080

 dense_206 (Dense)           (None, 50)                4050

 dense_207 (Dense)           (None, 30)                1530

 dense_208 (Dense)           (None, 1)                 31

=================================================================
Total params: 9,691
Trainable params: 9,691
Non-trainable params: 0
_________________________________________________________________
```

```
268/268 - 27s - loss: 0.5490 - accuracy: 0.7325 - 27s/epoch - 100ms/step
Loss: 0.548992931842804, Accuracy: 0.732478141784668
```

Despite modifying the parameters, we didn't observe a significant increase in performance from the initial model, and it does not meet the target 75% accuracy criteria. To improve the accuracy of our model, we attempted a systematic approach by following **Optimizing Neural Networks** and iteratively changing one model parameter at a time while holding others fixed, and then combining the parameters which generated the highest accuracy in each iterative search.

The results were the following:

Parameter	Search Options	Optimal Value	Loss	Accuracy
Training Duration (epochs)	50, 100, 200, 300	100	0.588	0.732
Architecture	All permutations with one to three hidden layers, i.e. [(10,), ..., (80,), (10, 30), (30, 10), ..., (80, 50), (10, 30, 50), (10, 50, 30), (30, 10, 50), ..., (80, 50, 30)	(80, 50, 30), three hidden layers with 80, 50, and 30 nodes.	0.561	0.740
Hidden Layer Activation Function	relu, tanh, selu, elu, exponential	relu and tanh	0.556	0.734
Number of Input Features	Bucket all combinations of APPLICATION_TYPE, CLASSIFICATION, INCOME_AMT		0.560	0.737

Combining all optimized model parameters, we retrained and tested to obtain the following testing loss and accuracy:

- Loss: 0.564
- Accuracy: 0.728

Summary

In summary, we presented a Deep Neural Network classification model that predicts loan applicant success from feature data contained in ***charity_data.csv*** with 73% accuracy. Unfortunately, our optimized model does not meet the 75% accuracy target, and the optimization methods employed here have not caused significant improvement.

Additional Optimization Methods

We can now consider other optimizing options to reach our more than 75% accuracy goal:

- Visualize the numerical feature variable ASK_AMT to find and remove potential noisy outliers.
- Carefully tune the parameters listed above iteratively and log optimal values when moving to reach an optimized model that meets our established goal.