Charity Funding Predictor

Author: Rosie Gianan

Objective:

The nonprofit foundation Alphabet Soup wants a tool to help select the applicants for funding with the best chance of success in their ventures. Their business team provided a charity_data.csv dataset containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. The dataset has a number of columns that capture metadata about each organization, such as:

- **EIN** and **NAME** - Identification columns
- **APPLICATION_TYPE** - Alphabet Soup application type
- **AFFILIATION** - Affiliated sector of industry
- **CLASSIFICATION** - Government organization classification
- **USE_CASE** - Use case for funding
- **ORGANIZATION** - Organization type
- **STATUS** - Active status
- **INCOME_AMT** - Income classification
- **SPECIAL_CONSIDERATIONS** - Special consideration for application
- **ASK_AMT** - Funding amount requested
- **IS_SUCCESSFUL** - Was the money used effectively

Solution:

Create the machine learning and neural network models to predict whether applicants will be successful if funded by Alphabet Soup. Using the features in the provided charity_data.csv dataset, build the neural network models using TensorFlow. The neural network model is to achieve a target predictive accuracy higher than 75%.

Processes:

1. Preprocess the data

Jupyter file: CharityFundingPredictor.ipynb

1.1 Load the dataset and identify the target and features columns.

1.1.1 Target variable: IS_SUCCESSFUL is the target variable which the values will be predicted using the using the rest of the variables in the dataset
IS_SUCCESSFUL int64; 1 is successful, 0 (unsuccessful)

1.1.2 Feature variables: Below are the feature variables which values will be used to predict the value of the target variable. (Note: This list excludes the 'EIN' and 'NAME' columns which are considered as not features columns in the creation of the original model.)

- APPLICATION_TYPE object
- AFFILIATION object

- CLASSIFICATION object
- USE_CASE object
- ORGANIZATION object
- STATUS int64
- INCOME_AMT object
- SPECIAL_CONSIDERATIONS object
- ASK_AMT int64

1.2 Drop the `EIN` and `NAME` columns from the input data because they are neither targets nor features.

1.3 Determine the number of unique values each column and the number of data points for unique values higher than 10. A cutoff value of 10 for columns 'APPLICATION_TYPE' and 'CLASSIFICATION' were used to replace the value to "Other".

1.4 Convert the categorical data into numeric using pd.get_dummies.

1.5 Split the preprocessed dataset into features and target arrays.

1.6 Scale the data using scikit-learn's `StandardScaler()`.

2. Compile, Train, and Evaluate the Model

Jupyter file: CharityFundingPredictor.ipynb

2.1 Create a neural network model with the following features:
- 1st hidden layer using activation function Relu (Rectified Linear Unit)
- 2nd hidden layer using activation function Sigmoid
- An output layer using activation function with units=1

2.2 Compile, train and evaluate the modules with 3 attempts. The results of these attempts didn't achieve the target accuracy of over 75%.

2.2.1 Attempt 1: Using 1st hidden layer = 8, 2nd hidden layer = 5 and epochs = 100. This attempt has an accuracy of 72.86%. The model for this attempt was saved in AlphabetSoupCharity.h5.

Parameters:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 576

 dense_1 (Dense)             (None, 5)                 45

 dense_2 (Dense)             (None, 1)                 6

=================================================================
Total params: 627
Trainable params: 627
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
268/268 - 1s - loss: 0.5501 - accuracy: 0.7286 - 520ms/epoch - 2ms/step
Loss: 0.5500602722167969, Accuracy: 0.7286297082901001
```

2.2.2 Attempt 2: Using 1st hidden layer = 16, 2nd hidden layer = 10 and epochs = 100. This attempt has an accuracy of 72.72% which is lower than the first attempt. The model for this attempt was saved in AlphabetSoupCharity_2.h5.

Parameters:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 16)                1152

 dense_4 (Dense)             (None, 10)                170

 dense_5 (Dense)             (None, 1)                 11

=================================================================
Total params: 1,333
Trainable params: 1,333
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
268/268 - 0s - loss: 0.5534 - accuracy: 0.7272 - 466ms/epoch - 2ms/step
Loss: 0.55335932970047, Accuracy: 0.7272303104400635
```

2.2.3 Attempt 3: Using 1st hidden layer = 32, 2nd hidden layer = 20 and epochs = 100. This attempt has an accuracy of 72.61% which is the lowest among the 3 attempts. The model for this attempt was saved in AlphabetSoupCharity_3.h5.

Parameters:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 32)                2304

 dense_7 (Dense)             (None, 20)                660

 dense_8 (Dense)             (None, 1)                 21

=================================================================
Total params: 2,985
Trainable params: 2,985
Non-trainable params: 0
```

Accuracy:

```
268/268 - 0s - loss: 0.5547 - accuracy: 0.7261 - 483ms/epoch - 2ms/step
Loss: 0.5546638369560242, Accuracy: 0.726064145565033
```

3. Optimize the original model to achieve the target predictive accuracy higher than 75%.

3.1 Preprocess the data similar to the step 1 with addition of column 'NAME' used as feature variable dropping only the 'EIN' column. A cutoff value of 10 for columns 'NAME', 'APPLICATION_TYPE' and 'CLASSIFICATION' were used to replace the value to "Other".

Jupyter file: AlphabetSoupCharity_Optimzation.ipynb

3.2 Optimize the neural network model by adding an additional hidden layer with the following features:

- 1st hidden layer using activation function Relu (Rectified Linear Unit)
- 2nd hidden layer using activation function Sigmoid
- 3rd hidden layer using activation function Sigmoid
- An output layer using activation function with units=1

3.3 Compile, train and evaluate the modules with 3 attempts. The results of the optimized model achieved the target accuracy of over 75%.

3.3.1 Attempt 1: Using 1st hidden layer = 8, 2nd hidden layer = 5, 3rd hidden layer = 5 and epochs = 100. This attempt has an accuracy of 78.4%. The model for this attempt was saved in AlphabetSoupCharity_Optimzation_1.h5.

Parameters:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 2360

 dense_1 (Dense)             (None, 5)                 45

 dense_2 (Dense)             (None, 5)                 30

 dense_3 (Dense)             (None, 1)                 6

=================================================================
Total params: 2,441
Trainable params: 2,441
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
268/268 - 1s - loss: 0.4491 - accuracy: 0.7840 - 538ms/epoch - 2ms/step
Loss: 0.4490695297718048, Accuracy: 0.7840233445167542
```

3.3.2 Attempt 2: Using 1st hidden layer = 16, 2nd hidden layer = 10, 3rd hidden layer = 10 and epochs = 100. This attempt has an accuracy of 78.61% which is the highest among the 3 attempts. The model for this attempt was saved in AlphabetSoupCharity_Optimzation_2.h5.

Parameters:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 16)                4720

 dense_5 (Dense)             (None, 10)                170

 dense_6 (Dense)             (None, 10)                110

 dense_7 (Dense)             (None, 1)                 11

=================================================================
Total params: 5,011
Trainable params: 5,011
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
268/268 - 0s - loss: 0.4533 - accuracy: 0.7861 - 462ms/epoch - 2ms/step
Loss: 0.453250527381897, Accuracy: 0.7861224412918091
```

3.3.3 Attempt 3: Using 1st hidden layer = 30, 2nd hidden layer = 15, 3rd hidden layer = 10 and epochs = 100. This attempt has an accuracy of 78.59% which is an insignificant difference from the 2nd attempt. The model for this attempt was save in AlphabetSoupCharity_Optimzation_3.h5.

Parameters:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 30)                8850

 dense_9 (Dense)             (None, 15)                465

 dense_10 (Dense)            (None, 10)                160

 dense_11 (Dense)            (None, 1)                 11

=================================================================
Total params: 9,486
Trainable params: 9,486
Non-trainable params: 0
_________________________________________________________________
```

Accuracy:

```
268/268 - 1s - loss: 0.4557 - accuracy: 0.7859 - 515ms/epoch - 2ms/step
Loss: 0.4557079076766968, Accuracy: 0.785889208316803
```

4. Summary

Original model: The result of the original model didn't achieve the target accuracy of over 75%. Three attempts were made using this model and the highest accuracy rate achieved was 72.86%. This model

excluded two columns (‘EIN’ and ‘NAME’) from the feature variables. This model uses two hidden layers using Relu and Sigmoid activation functions.

Optimized model: The result of the optimized model achieved the target accuracy of over 75%. Three attempts were made using this model and the highest accuracy rate achieved was 78.61%. This model excluded only the ‘EIN’ from the feature variables. An additional layer was added to this model using Sigmoid activation function.

Conclusion and Recommendation:

The three attempts were made using the original model and the optimized model. Using various combinations of the hidden layer for each model didn’t result in a significant difference in accuracy. However, by adding ‘NAME’ to the feature columns and an additional hidden layer to the optimized model resulted in a significant increase in the accuracy achieving the target rate of over 75%.

For the given data for this analysis, to achieve the target rate of over 75%, I recommend not dropping too many feature variables and limit the number of data being replaced as “Other”.