Module 21

Alphabet Soup Charity Funding Success Predictor

Overview

The aim of the module is to design a machine learning tool that can assist the foundation Alphabet Soup to find candidates with the best chance of success in their ventures. To develop our machine learning tools, we used neural network, keras and t-SHE. Our data was processed in jupyter notebook and google colab.

Results

Data Preprocessing

We removed EIN and Name columns from the dataframe and considered the rest of them. The target column was [IS_Successful] with values of 0 and 1 for success and failure. We binned Application_type and Classification and converted all values to numerical values using pd.get dummies function.

```python
# Drop the non-beneficial ID columns, 'EIN' and 'NAME'.
application_df = application_df.drop(columns=['EIN', 'NAME'], axis=1)
application_df.head()
```

	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CASE	ORGANIZATION	STATUS	INCOME_AMT	SPECIAL_CONSIDERATIONS	ASK_AMT	IS_
0	T10	Independent	C1000	ProductDev	Association	1	0	N	5000	
1	T3	Independent	C2000	Preservation	Co-operative	1	1-9999	N	108590	
2	T5	CompanySponsored	C3000	ProductDev	Association	1	0	N	5000	
3	T3	CompanySponsored	C2000	Preservation	Trust	1	10000-24999	N	6692	
4	T3	Independent	C1000	Heathcare	Trust	1	100000-499999	N	142590	

```python
# Determine the number of unique values in each column.
application_df.nunique()
```

```
APPLICATION_TYPE            17
AFFILIATION                  6
CLASSIFICATION              71
USE_CASE                     5
ORGANIZATION                 4
STATUS                       2
INCOME_AMT                   9
SPECIAL_CONSIDERATIONS       2
ASK_AMT                   8747
IS_SUCCESSFUL                2
dtype: int64
```

```python
# Look at APPLICATION_TYPE value counts for binning
application_counts=application_df['APPLICATION_TYPE'].value_counts()
application_counts
```

```
APPLICATION_TYPE
T3     27037
T4      1542
T6      1216
T5      1173
T19     1065
T8       737
T7       725
T10      528
T9       156
T13       66
T12       27
T2        16
T25        3
T14        3
T29        2
T15        2
T17        1
Name: count, dtype: int64
```

```python
# Choose a cutoff value and create a list of application types to be replaced
# use the variable name `application_types_to_replace`
application_types_to_replace = list(application_counts[application_counts<500].index)

# Replace in dataframe
for app in application_types_to_replace:
    application_df['APPLICATION_TYPE'] = application_df['APPLICATION_TYPE'].replace(app,"Other")

# Check to make sure binning was successful
application_df['APPLICATION_TYPE'].value_counts()
```

Compiling, Training, and Evaluating the Model

Four models were created. The first model had 2 layers, 30 neurons in the first layer and 10 in the second. This model showed an accuracy of 73.21%.

```python
# 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 = 30
hidden_nodes_layer2 = 10
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()
```

```
WARNING:tensorflow:From C:\Users\alipe\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is d
eprecated. Please use tf.compat.v1.get_default_graph instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 30)                1500

 dense_1 (Dense)             (None, 10)                310

 dense_2 (Dense)             (None, 1)                 11

=================================================================
Total params: 1821 (7.11 KB)
Trainable params: 1821 (7.11 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

For the first optimization, 2 second hidden layers were added with 10 and 30 neurons and it depreciated the accuracy to 72.97%.

```python
# 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 = 30
hidden_nodes_layer2 = 10
hidden_nodes_layer3 = 30
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'))

nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation='relu'))

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

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

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 30)                1500

 dense_3 (Dense)             (None, 10)                310

 dense_4 (Dense)             (None, 30)                330

 dense_5 (Dense)             (None, 1)                 31

=================================================================
Total params: 2171 (8.48 KB)
Trainable params: 2171 (8.48 KB)
Non-trainable params: 0 (0.00 Byte)
```

For the second optimization I added 4 hidden layers with 10,20,30 and 30 neurons. The accuracy further decreased to 72.76%.

```
# 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 = 10
hidden_nodes_layer2 = 20
hidden_nodes_layer3 = 30
hidden_nodes_layer4 = 30
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'))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer3, activation='relu'))

# Fourth hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer4, activation='relu'))

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

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

```
WARNING:tensorflow:From C:\Users\alipe\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is d
eprecated. Please use tf.compat.v1.get_default_graph instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                500

 dense_1 (Dense)             (None, 20)                220

 dense_2 (Dense)             (None, 30)                630

 dense_3 (Dense)             (None, 30)                930

 dense_4 (Dense)             (None, 1)                 31

=================================================================
Total params: 2311 (9.03 KB)
Trainable params: 2311 (9.03 KB)
Non-trainable params: 0 (0.00 Byte)
```

For the third optimization I kept 2 layers but increased neurons to 100 and 50. This slightly increased accuracy to 72.77%.

```python
# 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 = 100
hidden_nodes_layer2 = 50
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()
```

```
WARNING:tensorflow:From C:\Users\alipe\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is d
eprecated. Please use tf.compat.v1.get_default_graph instead.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               5000

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 10101 (39.46 KB)
Trainable params: 10101 (39.46 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Summary

We built a machine learning model to assess an applicants eligibility to receive funding for ventures. Our models were able to obtain a maximum accuracy of 73%. Adding additional layers or neurons to the layers had little effect on the accuracy.

We could try to use random forests. Random forest has an emphasis on misclassified observations or fitting new learners to minimize the mean-squared error between the observed response and the aggregated prediction of all previously grown learners which could work for our model.