Neural Network Report

Overview of the analysis:

The nonprofit foundation Alphabet Soup wanted a tool to help them select applicants for funding. They need a binary classifier to predict whether applicants will succeed if alphabet soup funds them.

Results: Using bulleted lists and images to support your answers, address the following questions.

Data Preprocessing

What variable(s) are the target(s) for your model?

*This data frame with columns for EIN, Name, Application type, classification, use case, organization, status, income amount, special considerations, ask amount, and if it is successful. We want to predict if applicants will succeed from this data.

```
application_df = pd.read_csv("Resources/charity_data.csv")
application_df.head()
```

Python

	EIN	NAME	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CASE	ORGANIZATION	STATUS	INCOME_AMT	SPECIAL_CONSIDERATIONS	ASK_AMT	IS_SUCCESSFUL
0	10520599	BLUE KNIGHTS MOTORCYCLE CLUB	T10	Independent	C1000	ProductDev	Association	1	0	N	5000	1
1	10531628	AMERICAN CHESAPEAKE CLUB CHARITABLE TR	T3	Independent	C2000	Preservation	Co-operative	1	1-9999	N	108590	1
2	10547893	ST CLOUD PROFESSIONAL FIREFIGHTERS	T5	CompanySponsored	C3000	ProductDev	Association	1	0	N	5000	0
3	10553066	SOUTHSIDE ATHLETIC ASSOCIATION	T3	CompanySponsored	C2000	Preservation	Trust	1	10000-24999	N	6692	1
4	10556103	GENETIC RESEARCH INSTITUTE OF THE DESERT	T3	Independent	C1000	Heathcare	Trust	1	100000-499999	N	142590	1

```
# Drop the non-beneficial ID columns, 'EIN' and 'NAME'.
```

What variable(s) are the features for your model?

* the two original variables in the model that we targeted were classification and Application Type. In the final optimization code, however, we used name and classification.

```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(counts[counts<500].index)
application_types_to_replace

# 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()
```

```
T3       27037
T4        1542
T6        1216
T5        1173
T19       1065
T8         737
T7         725
T10        528
Other      276
Name: APPLICATION_TYPE, dtype: int64
```

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

```
C1000    17326
C2000     6074
C1200     4837
C3000     1918
C2100     1883
         ...
C4120        1
C8210        1
C2561        1
C4500        1
C2150        1
Name: CLASSIFICATION, Length: 71, dtype: int64
```

```
# Choose a cutoff value and create a list of names to be replaced
# use the variable name `names_to_replace`
names_to_replace = list(counts[counts<10].index)
names_to_replace

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

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

```
Other                                    21022
PARENT BOOSTER USA INC                    1260
TOPS CLUB INC                              765
UNITED STATES BOWLING CONGRESS INC         700
WASHINGTON STATE UNIVERSITY                492
                                          ...
CASCADE 4-H FOUNDATION                      10
FREE & ACCEPTED MASONS OF WASHINGTON        10
NEW MEXICO GARDEN CLUBS INC                 10
NATIONAL ASSOCIATION OF HISPANIC NURSES     10
UNION OF CALIFORNIA STATE WORKERS           10
Name: NAME, Length: 223, dtype: int64
```

```
# Look at CLASSIFICATION value counts for binning
classificationbinning = application_df['CLASSIFICATION'].value_counts()
classificationbinning
```

```
C1000    17326
C2000     6074
C1200     4837
C3000     1918
C2100     1883
         ...
C4120        1
C8210        1
C2561        1
```

What variable(s) should be removed from the input data because they are neither targets nor features?

*Original model we removed EIN number and Name, but in optimization 2 we only get rid of EIN number

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

```
# Drop the non-beneficial ID columns, just dropping 'EIN' this time.
application_df = application_df.drop(columns = ['EIN'])
```

Compiling, Training, and Evaluating the Model

How many neurons, layers, and activation functions did you select for your neural network model, and why?

* Originally we went with 2 layers but when we optimize we did 3. Nodes we did 80 and 30 for the original one. For the final optimization, we did 3 layers with 7, 14, and 21 nodes.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_features = X_train_scaled.shape[1]
hidden_nodes1=80
hidden_nodes2=30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='relu'))

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

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

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_features = X_train_scaled.shape[1]
hidden_nodes1=7
hidden_nodes2=14
hidden_nodes3=21

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='relu'))

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes3, activation='relu'))

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

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

Were you able to achieve the target model performance?

- On the final optimization, we got up to 78%

What steps did you take in your attempts to increase model performance?

- In the first optimization, we added another neuron and more epochs. On the second optimization, we used the name as a feature and did not drop it which helped it achieve 78%

```
    # 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: 0.4723 - accuracy: 0.7736 - 309ms/epoch - 1ms/step
Loss: 0.47229671478271484, Accuracy: 0.7736443281173706
```

Summary: Summarize the overall results of the deep learning model. Include a recommendation for how a different model could solve this classification problem, and then explain your recommendation.

- Several layers should be considered, so it can continue to predict and classify information based on the model.