Report

The purpose of this analysis is to help Alphabet Soup on creating a tool to select the applicants for funding with the best chance of success in their ventures. Applying different features and using machine learning I would be creating a model that would help predict whether applicants will be successful if funded by Alphabet Soup.

- Data Preprocessing
    - What variable(s) are the target(s) for your model? The target variable for my analysis was 'IS SUCCESSFULL' with the values 0 and 1
    - What variable(s) are the features for your model? The features variables for the first part of the assignment were APPLICATION_TYPE AND CLASSIFICATION, for the optimization part the features also included the NAME column.
    - What variable(s) should be removed from the input data because they are neither targets nor features? The columns that were removed from the dataset were EIN and NAME for the first model created. for the optimization model just the EIN column was removed

    First model:

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

    Optimization model:

```
# Drop the non-beneficial ID columns, 'EIN'
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? In the provided code, the neural network model is constructed using TensorFlow's Keras API. Here are the details regarding the number of neurons, layers, and activation functions: The neural network model consists of 3 layers with a total of 229 neurons (128 + 100 + 1) for the first model created. The first and second hidden layers use the ReLU activation function, while the output layer uses the sigmoid activation function. These choices balance model complexity, computational efficiency, and suitability for binary classification tasks.

```
import tensorflow as tf

input_shape = len(X_train[0])

# Define the model
nn = tf.keras.models.Sequential()

#first hidden layer
nn.add(tf.keras.layers.Dense(units=128, activation='relu', input_dim=input_shape))
nn.add(tf.keras.layers.Dense(units=100, activation='relu'))
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
nn.summary()
```

- Were you able to achieve the target model performance? yes
- What steps did you take in your attempts to increase model performance?

I was able to get 78% accuracy adjusting some of the layers and features on the model as well as including the 'NAME' variables. For the optimization model to increase model performance 3 more layers were added to improve the accuracy where the total neurons were 231. For the activation layer using the feature relo while the output layers the sigmoid function were used.

```
import tensorflow as tf

# Assume X_train is already defined and preprocessed
input_shape = len(X_train[0])

# Define the model
nn = tf.keras.models.Sequential()

# layer
nn.add(tf.keras.layers.Dense(units=100, activation='relu', input_dim=input_shape))
nn.add(tf.keras.layers.Dense(units=80, activation='sigmoid'))
nn.add(tf.keras.layers.Dense(units=30, activation='sigmoid'))
nn.add(tf.keras.layers.Dense(units=20, activation='sigmoid'))
nn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
nn.summary()
```

Summary:

Given the initial success with the deep learning model, exploring ensemble methods combining deep learning with tree-based algorithms like Random Forest or Gradient Boosting could be highly beneficial. These methods often provide better performance by capturing different aspects of the data and reducing the risk of overfitting.