Report on the Neural Network Model:
Alphabet Soup Charity Funding Predictions

Analysis:

The purpose of this analysis was to develop a machine learning module that is able to predict whether an organization funded by Alphabet Soup will turn out successful or not. We would use a binary classifier after compiling and training the module to help make predictions on its likelihood of success and accuracy ratings.

Based on historical data in a CSV containing over 34,000 organizations including each of their application types, affiliation, classifications, income, and funding requests, our goal is to interpret any strong patterns that would lead into successful and unsuccessful outcomes.

Results:

Data Preprocessing:

- **Target Variable:**
  - Our only target variable was "**IS_SUCCESSFUL**, where "**1**" would be marked as successfully utilizing the organization's funding, "**0**" where it was unsuccessful.
- **Feature Variables:**
  - The features used for the model included:
    - **APPLICATION_TYPE**; type of application submitted to Alphabet Soup.
    - **AFFILIATION**; affiliated sector of industry.
    - **CLASSIFICATION**; government organization classification.
    - **INCOME_AMT**; income classification.
    - **ASK_AMT**; the amount of funding requested.
- **Removed Variables:**
  - The following variables were removed from the input data as compared to other columns, these did not provide as much importance and could negatively affect the model's result:
    - **EIN**; Employer Identification Number (unique identifier for organizations).
    - **NAME**; organization's name.
  - After further tests, we later removed more columns in an attempt to further improve our results:
    - **USE_CASE**; use case for funding.
    - **ORGANIZATION**; organization's type.
    - **STATUS**; active status of organization.
    - **SPECIAL_CONSIDERATIONS**; special considerations for application.

Compiling, Training, and Evaluating the Model:

- **Model Architecture:**
  - The neural network model consists of the following structure:
    - **Input Layer**: takes in all feature variables after preprocessing.
    - **First Hidden Layer**: 128 neurons with the ReLU activation function.
    - **Second Hidden Layer**: 64 neurons with the ReLU activation function.
    - **Third Hidden Layer**: 32 neurons with the ReLU activation function.
    - **Fourth Hidden Layer**: 16 neurons with the ReLU activation function.
    - **Output Layer**: one neuron with the sigmoid activation function for binary classification.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=256, activation='relu', input_dim=X_train_scaled.shape[1]))  #1st time: 80  #2nd time: 128   #final: 256

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=128, activation='relu'))  #1st time: 30   #2nd time: 30  #final: 128

# Third hidden layer
nn.add(tf.keras.layers.Dense(units=64, activation='relu'))  #3rd time: implemented a third layer at 25   #final: 64

# Fourth hidden layer                                      #(added in final optimization)
nn.add(tf.keras.layers.Dense(units=32, activation='relu'))

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

- **Model Performance:**
  - **Final Model Accuracy**: the model achieved an accuracy of approximately **72.68%** on the test data.
    - Out of all attempts, the highest ever was **72.94%**.
  - **Model Loss**: The final loss of the model was **0.6004**, indicating significant room for improvement in model predictions.

```
#INITIAL FILE

#1st Results:                                                  #2nd Results:
#215/215 - 1s - 3ms/step - accuracy: 0.7248 - loss: 0.5640     #215/215 - 0s - 2ms/step - accuracy: 0.7271 - loss: 0.5712
#Loss: 0.5640185475349426, Accuracy: 0.724781334400177         #Loss: 0.5712478756904602, Accuracy: 0.7271137237548828

#3rd Results:                                                  #4th Results:
#215/215 - 0s - 1ms/step - accuracy: 0.7249 - loss: 0.6128     #215/215 - 0s - 2ms/step - accuracy: 0.7206 - loss: 0.5811
#Loss: 0.6128414869308472, Accuracy: 0.7249271273612976        #Loss: 0.5810864567756653, Accuracy: 0.7205539345741272

#5th Results:
#215/215 - 0s - 1ms/step - accuracy: 0.7294 - loss: 0.5564
#Loss: 0.5564342141151428, Accuracy: 0.7294460535049438

#OPTIMIZATION FILE

#6th Results:                                                  #7th Results (final):
#215/215 - 0s - 2ms/step - accuracy: 0.7239 - loss: 0.5812     #215/215 - 0s - 2ms/step - accuracy: 0.7268 - loss: 0.6004
#Loss: 0.5811856985092163, Accuracy: 0.7239066958427429        #Loss: 0.6004123091697693, Accuracy: 0.7268221378326416
```

- **Steps Taken to Improve Model Performance:**
    - Dropped less significant features to reduce noise and improve model training.
    - Added more neurons and layers to the model to improve learning capacity.
    - Experimented with different batch sizes and epochs to find optimal training conditions.

Summary:

The deep learning model developed for Alphabet Soup achieved an accuracy rating of 72.68% in predicting the success of funding recipients. Despite several attempts in improving the model's performance, it did not meet the targeted 75% accuracy. Different approaches would be required if we were to optimize the model in order to achieve a target predictive accuracy equal or higher than 75%.

Recommendations:

- Consider using different machine learning models such as **Random Forests** or **Gradient Boosting Machines (GBM)**. These models typically handle non-linear relationships and interactions more effectively than a single neural network model. They also provide feature importance scores, to help understand which variables contributed the most to the predictions to help narrow which part of the data requires polishing.

- Utilize hyperparameter optimization techniques, such as **Grid Search** or **Random Search**, to find the best combination of hyperparameters for the chosen model.

With just these alternative approaches, Alphabet Soup can further refine its predictive modeling capabilities with better results, ultimately improving the decision-making process for funding allocations.