Alphabet Soup Nonprofit

Overview: This analysis for the nonprofit foundation Alphabet Soup was generated in order to select applicants with the highest chance of success in their aspirations. The foundation has provided funds for over 34,000 organizations and will be used to determine the success rate if funded by the foundation.

Results: The data processing began with the removal of any irrelevant columns in the CSV. Those columns were EIN and NAME columns, the identification columns of the original data provided. In the model the target is the dependent variable “IS_SUCESSFUL” as it stands for the variable the model is trying to predict. The remainder of the columns creates the variables of the features for the model. Including application type, affiliation , classification columns that contain factors that potentially influence the success rate. In the model I created I proceeded to remove the ‘SPECIAL_CONSIDERATION’ to provide some level of equal opportunity to all applicants in addition to the identification columns .

```
[ ] # Split our preprocessed data into our features and target arrays

    y = application_df['IS_SUCCESSFUL']
    X = application_df.drop(columns=['IS_SUCCESSFUL'])

    # Split the preprocessed data into a training and testing dataset
    X_train, X_test, y_train, y_test = train_test_split(X,y,random_state = 42)
```

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

```
# Drop column 'SPECIAL_CONSIDERATIONS' to create some level of equality.
application_df = application_df.drop(columns=['SPECIAL_CONSIDERATIONS'], axis=1)
application_df.head()
```

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

Compiling,Training, and Evaluating the Model: During my experimental process a third layer was added to serve as an additional filter for the data. Starting off with a wider hidden layer containing 100 nodes and the node count getting smaller within the last two hidden layers. One final adjustment to my model was the increase in epochs to allow the model to better understand as it doubles the passing process.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
# The first hidden layer  is wider with 1000 nodes, second hidden layer has 60 nodes, and third 20.
number_input_features = len(X_train.columns)
hidden_nodes_layer1 =  100
hidden_nodes_layer2 = 60
hidden_nodes_layer3 = 20

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.Input(shape=(number_input_features,)))
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, 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"))


# 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_3 (Dense)	(None, 100)	4,900
dense_4 (Dense)	(None, 60)	6,060
dense_5 (Dense)	(None, 20)	1,220
dense_6 (Dense)	(None, 1)	21

Total params: 12,201 (47.66 KB)

Unfortunately, the changes in the attempts to improve the model were slightly below the desired 75% with an 73% accuracy. Still providing a 1% increase from the evaluation.

```
# 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 - 1s - 2ms/step - accuracy: 0.7301 - loss: 0.5854
Loss: 0.5854431986808777, Accuracy: 0.7301457524299622
```