Deep Learning Analysis:

The purpose with this model was to see whether or not the individual who received the money would be successful or not. Thus, our target variable was “IS_SUCCESFUL”. The columns needed to be deleted were identifiers such as “EIN” and “NAME” while the rest were used as features. Before using get_dummies, I created bins for “CLASSIFICATION” and “APPLICATION_TYPE,” which allowed me to reduce the amount of columns needed.

```
# Split our preprocessed data into our features and target arrays
y = X_dummies['IS_SUCCESSFUL'].values
X = X_dummies.drop(columns = ['IS_SUCCESSFUL']).values
# 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)
```

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

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

At first, my model only contained two hidden layers, both of them being relu, while the output was sigmoid.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
num_input = len(X_train_scaled[0])
num_first = len(X_train_scaled[0])*2
num_second = 50

nn = tf.keras.models.Sequential()

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

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

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

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

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                3280
_________________________________________________________________
dense_1 (Dense)              (None, 50)                4050
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 7,381
Trainable params: 7,381
Non-trainable params: 0
```

However, this model only received an accuracy score of 73.06 percent. Thus, I decided to increase the amount of hidden layers by two with the same activation, relu.

```
num_input = len(X_train_scaled[0])
num_first = len(X_train_scaled[0])*2
num_second = 50

nn = tf.keras.models.Sequential()

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

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

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

nn.add(tf.keras.layers.Dense(units = num_second, 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, 80)                3280
_________________________________________________________________
dense_4 (Dense)              (None, 50)                4050
_________________________________________________________________
dense_5 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_6 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 51
=================================================================
Total params: 12,481
Trainable params: 12,481
Non-trainable params: 0
```

This model resulted in an accuracy score of 72.99, which was slightly worse than the previous model. Therefore, for my last attempt, I decided to switch the type of activation since increasing it does not seem to increase the accuracy.

```
num_input = len(X_train_scaled[0])
num_first = len(X_train_scaled[0])*2
num_second = 50

nn = tf.keras.models.Sequential()

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

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

nn.add(tf.keras.layers.Dense(units = num_second, activation='sigmoid'))

nn.add(tf.keras.layers.Dense(units = num_second, activation='sigmoid'))

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

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

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 80)                3280
_________________________________________________________________
dense_9 (Dense)              (None, 50)                4050
_________________________________________________________________
dense_10 (Dense)             (None, 50)                2550
_________________________________________________________________
dense_11 (Dense)             (None, 50)                2550
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 51
=================================================================
Total params: 12,481
Trainable params: 12,481
Non-trainable params: 0
_________________________________________________________________
```

However, the outcome of this model was the same as before with a slightly worse score of 72.85. Unfortunately, I was unable to reach the target score of above 75. To fix this in the future, I could drop certain columns that may not hold as much importance and thus decrease the amount of noise the machine has to go through. Another way could be to increase the amount of bins to make a broader statement.