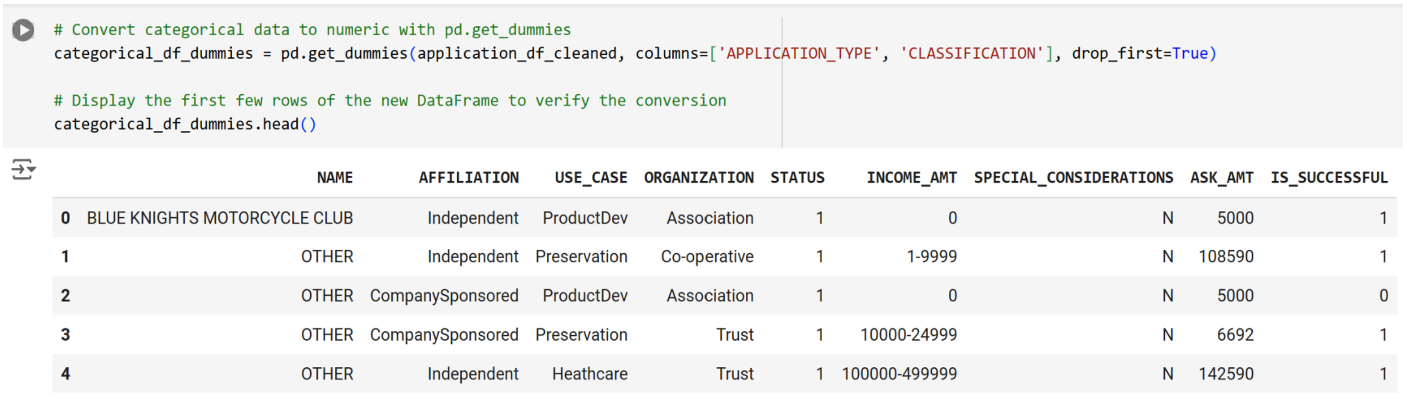
Neural Network Model Report

By Christopher Swires

Overview

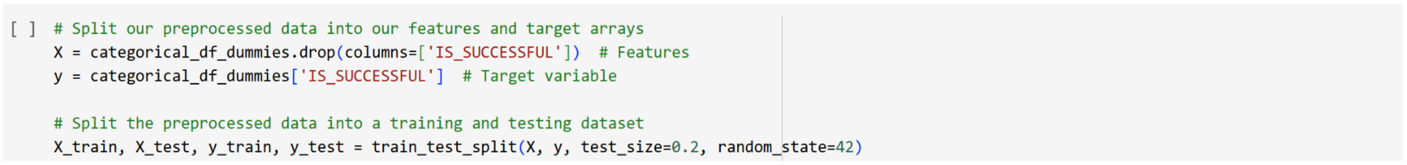
The purpose of this assignment was to design a neural network that could determine the success of nonprofit funding. In order to do that, I had to preprocess the data by reading the CSV file into a dataframe and then filtering it by dropping columns and adding bins to miscellaneous categories within the dataframe. Once the data was preprocessed, I had to ensure that the neural network would return at least 75% target predictive accuracy. Certain factors that affected the target predictive accuracy were the amount of hidden layers and neurons, certain activation functions, and certain columns in the data frame. Once I had analyzed and tested out which components would have the biggest impact on target predictive accuracy, I was able to see which specific components were the most beneficial for a 75% target predictive accuracy score.

Results



```python
# Convert categorical data to numeric with pd.get_dummies
categorical_df_dummies = pd.get_dummies(application_df_cleaned, columns=['APPLICATION_TYPE', 'CLASSIFICATION'], drop_first=True)

# Display the first few rows of the new DataFrame to verify the conversion
categorical_df_dummies.head()
```

	NAME	AFFILIATION	USE_CASE	ORGANIZATION	STATUS	INCOME_AMT	SPECIAL_CONSIDERATIONS	ASK_AMT	IS_SUCCESSFUL
0	BLUE KNIGHTS MOTORCYCLE CLUB	Independent	ProductDev	Association	1	0	N	5000	1
1	OTHER	Independent	Preservation	Co-operative	1	1-9999	N	108590	1
2	OTHER	CompanySponsored	ProductDev	Association	1	0	N	5000	0
3	OTHER	CompanySponsored	Preservation	Trust	1	10000-24999	N	6692	1
4	OTHER	Independent	Heathcare	Trust	1	100000-499999	N	142590	1



```python
[ ] # Split our preprocessed data into our features and target arrays
    X = categorical_df_dummies.drop(columns=['IS_SUCCESSFUL'])  # Features
    y = categorical_df_dummies['IS_SUCCESSFUL']  # Target variable

    # Split the preprocessed data into a training and testing dataset
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Preprocessing

Target variable column:

- “Is_Successful”

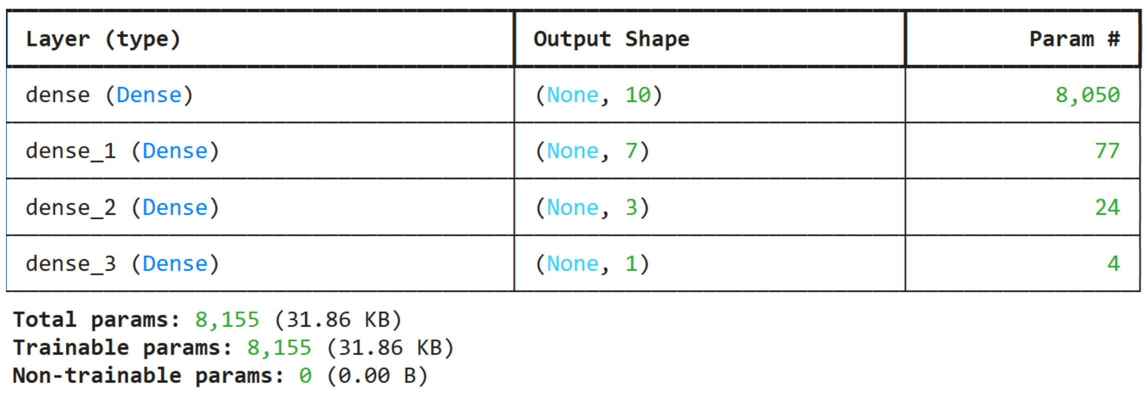
Target feature columns:

- “Name”
- “Affiliation”
- “Classification”
- “Use_Case”
- “Organization”
- “Status”
- “Income_Amt”
- “Special_Considerations”
- “Ask_Amt”
- “Is_Succesful”

Dropped column:

- “EIN”

Compiling, Training and Evaluating



```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ dense (Dense)                        │ (None, 10)                  │           8,050 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 7)                   │              77 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 3)                   │              24 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 1)                   │               4 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 8,155 (31.86 KB)
 Trainable params: 8,155 (31.86 KB)
 Non-trainable params: 0 (0.00 B)
```

Number of neurons added:

- Layer 1 - 10 neurons
- Layer 2 - 7 neurons
- Layer 3 - 3 neurons

Number of layers added:

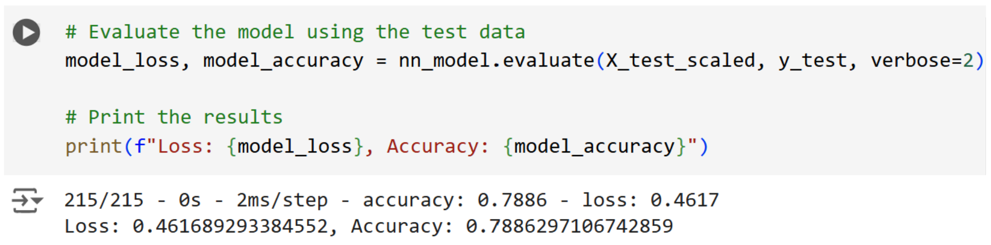
- Third layer added

Activation functions used:

- Relu
- Sigmoid

Achieving Target Performance

Was I able to achieve target performance? **Yes!**



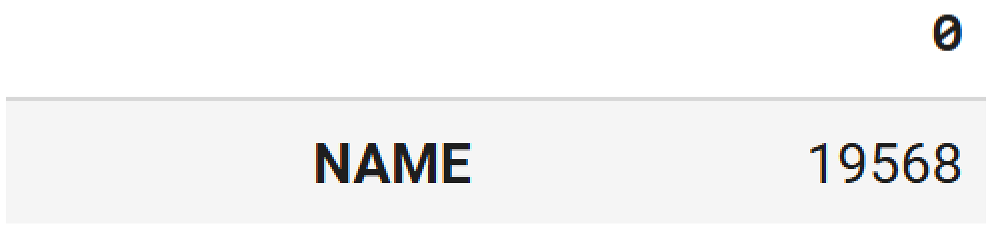
```
# Evaluate the model using the test data
model_loss, model_accuracy = nn_model.evaluate(X_test_scaled, y_test, verbose=2)

# Print the results
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
215/215 - 0s - 2ms/step - accuracy: 0.7886 - loss: 0.4617
Loss: 0.461689293384552, Accuracy: 0.7886297106742859
```

What steps did I take to achieve target performance?

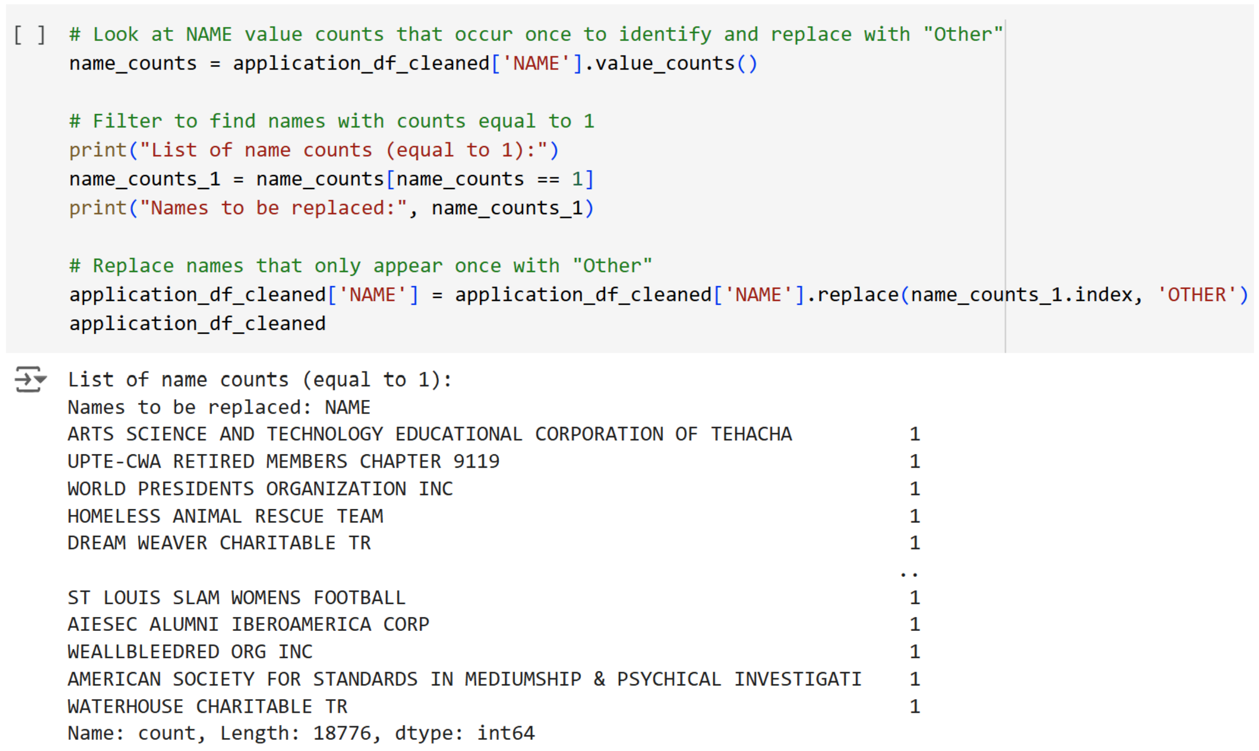
- I started by adding the third layer and adjusting the number of neurons
- I tested the accuracy and did not reach the target performance of 75%
    - I removed the “drop” command from the “Name” column
    - I used a function to return the names that only occurred once
    - I replaced the values of the names occurring once with “OTHER”
    - I added the modified “NAME” column to target features

Number of unique values in “NAME” column



	0
NAME	19568

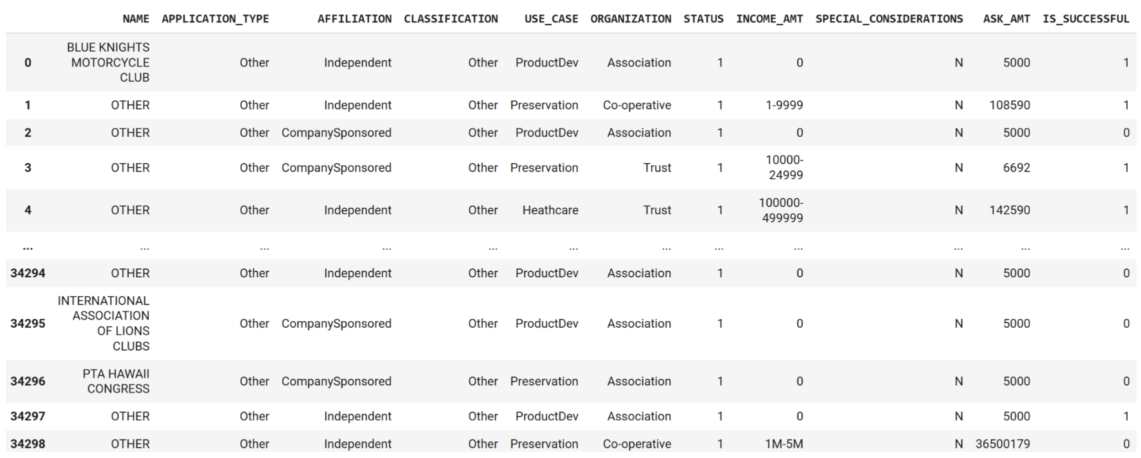
Code used to modify and incorporate “NAME” column



```
[ ]  # Look at NAME value counts that occur once to identify and replace with "Other"
     name_counts = application_df_cleaned['NAME'].value_counts()

     # Filter to find names with counts equal to 1
     print("List of name counts (equal to 1):")
     name_counts_1 = name_counts[name_counts == 1]
     print("Names to be replaced:", name_counts_1)

     # Replace names that only appear once with "Other"
     application_df_cleaned['NAME'] = application_df_cleaned['NAME'].replace(name_counts_1.index, 'OTHER')
     application_df_cleaned
```

```
List of name counts (equal to 1):
Names to be replaced: NAME
ARTS SCIENCE AND TECHNOLOGY EDUCATIONAL CORPORATION OF TEHACHA          1
UPTE-CWA RETIRED MEMBERS CHAPTER 9119                                   1
WORLD PRESIDENTS ORGANIZATION INC                                       1
HOMELESS ANIMAL RESCUE TEAM                                             1
DREAM WEAVER CHARITABLE TR                                              1
                                                                       ..
ST LOUIS SLAM WOMENS FOOTBALL                                           1
AIESEC ALUMNI IBEROAMERICA CORP                                         1
WEALLBLEEDRED ORG INC                                                   1
AMERICAN SOCIETY FOR STANDARDS IN MEDIUMSHIP & PSYCHICAL INVESTIGATI    1
WATERHOUSE CHARITABLE TR                                                1
Name: count, Length: 18776, dtype: int64
```

Dataframe with modified “NAME” column



	NAME	APPLICATION_TYPE	AFFILIATION	CLASSIFICATION	USE_CASE	ORGANIZATION	STATUS	INCOME_AMT	SPECIAL_CONSIDERATIONS	ASK_AMT	IS_SUCCESSFUL
0	BLUE KNIGHTS MOTORCYCLE CLUB	Other	Independent	Other	ProductDev	Association	1	0	N	5000	1
1	OTHER	Other	Independent	Other	Preservation	Co-operative	1	1-9999	N	108590	1
2	OTHER	Other	CompanySponsored	Other	ProductDev	Association	1	0	N	5000	0
3	OTHER	Other	CompanySponsored	Other	Preservation	Trust	1	10000-24999	N	6692	1
4	OTHER	Other	Independent	Other	Heathcare	Trust	1	100000-499999	N	142590	1
...	...	...	...	...	...	...	...	...	...	...	...
34294	OTHER	Other	Independent	Other	ProductDev	Association	1	0	N	5000	0
34295	INTERNATIONAL ASSOCIATION OF LIONS CLUBS	Other	CompanySponsored	Other	ProductDev	Association	1	0	N	5000	0
34296	PTA HAWAII CONGRESS	Other	CompanySponsored	Other	Preservation	Association	1	0	N	5000	0
34297	OTHER	Other	Independent	Other	ProductDev	Association	1	0	N	5000	1
34298	OTHER	Other	Independent	Other	Preservation	Co-operative	1	1M-5M	N	36500179	0

Summary

The objective of this neural network model was to take preprocessed code from a starter code file and modify it in order to see if the target performance of 75% accuracy could be achieved. My main takeaway is that the “NAME” column is a vital component for achieving accuracy as it has many values, most of which only occur once. By bringing the “NAME” column back into the dataframe and renaming the once occurring values as “OTHER”, I was able to make the dataframe more organized while incorporating important data that was used as a feature to determine success.

Although it certainly can help to add more layers and neurons, the difference is minuscule - I barely noticed any increase in accuracy after I had added the third layer as well as increasing the neurons. Therefore, I would recommend calculating how many unique values there are for each of the columns before dropping any columns and adding any layers or neurons. If there are a high number of unique values, I recommend running code to determine how many of them only occur once so that they can be categorized as “OTHER”. I got the idea from the starter code where I was instructed to replace all of the values in “APPLICATION_TYPE” with “OTHER”.

Overall, I learned that in order to build a neural network with high accuracy, you need to have as much categorical data as possible, and only columns that have all unique values should be dropped. The “EIN” column is a prime example of a value that is not relevant to the target column (“IS_SUCCESSFUL”) as they are not able to be categorized as they are all unique values. Columns that have values that are both repeating and unique are beneficial to have in the dataframe as they can be used to categorize data, and the unique values can be categorized as “OTHER”.