Overview of the Analysis

In this analysis I was tasked with helping to create a tool that can help a nonprofit foundation select applicants for funding that have the best chance of success. Using machine learning and neural networks I worked to train and run models that will make it easier for the foundation to have confidence in where their funding is going.

This data contains more than 34,000 organizations that have received funding from the foundation in the recent years. In the dataset there are metadata columns about each organization including name, application type, use case, asking amount, and several others. With these columns my task was to select the columns or features that would best predict which organizations will be successful if funded by the foundation based on the metadata gathered. In this analysis I was trying to predict IS_SUCCESSFUL, which was determined if an organization had success in their venture after being funded from the foundation. By training the models on this variable the foundation should be able to predict what future organizations will be successful if funded.

Before starting the machine learning process, I needed to do some preprocessing and data cleaning. First, I checked for null values as well as dtypes. Next, I initially dropped both the EIN and the NAME column but after running a few models I decided to keep the name column to see if it improved my models. After dropping the EIN column, I began to look at the value counts to see what kind of bucketing needed to be done to balance out the data. I ended up bucketing the NAME, APPLICATION TYPE, CLASSIFICATION, ORGANIZATION, USE CASE, and AFFILIATION columns to try and help with some of the imbalance issues. After the data cleaning I then used the pd.get_dummies function to convert the categorical data into numeric values.

Once the data was preprocessed, I split the data into the features and the target array. I then trained the model and scaled the data. Once the models were trained, I started to set up my neural networks.

Results

For all the models the target is IS_SUCCESSFUL and the features are the remaining columns. Initially I only dropped the NAME and EIN because they didn't seem to add any predictive value to the model.

```
# Drop the non-beneficial ID columns, 'EIN' and 'NAME'.
#creating a list of columns to drop
cols_to_drop = ['EIN', 'NAME']
application_df.drop(columns = cols_to_drop, inplace =True)
application_df.head()
```

But, after running the first three models it was clear that I needed to do some more data

cleaning because the accuracy was still low. I made a copy of the data frames and then dropped the SPECIAL_CONSIDERATIONS and STATUS columns due to severe imbalance in hopes that it would improve the results of the model.

```
# Droping special considerations and status due to the severe
#imbalance to try and imporove the model
cols_to_drop = ['SPECIAL_CONSIDERATIONS', 'STATUS']
app_copy.drop(columns = cols_to_drop, inplace =True)
app_copy.head()
```

Model 1:

For this first model I set up the initial neural network and only added one layer with a few neurons to get a feel on how the model would perform.

```
268/268 - 1s - 3ms/step - accuracy: 0.7251 - loss: 0.5559
Loss: 0.5559373497962952, Accuracy: 0.7251312136650085
```

- This Initial model was a good starting point to see what I should add or change to improve the accuracy.
- For this model I used the relu activation for the unput layer and sigmoid for the output layer
- Only used 20 epochs to again, get an idea of how the model would perform
- This first layer only had 7 neurons

Model 2:

For this model I decided to add another layer and more neurons to the layers in hopes that another layer and more neurons would help increase the accuracy for the model.

```
268/268 - 1s - 2ms/step - accuracy: 0.7272 - loss: 0.5564
Loss: 0.5564322471618652, Accuracy: 0.7272303104400635
```

- This model had two layers, the first with 30 neurons and the second with 15 neurons.
- I also changed the activation for the second input layer to tanh instead of both layers using the relu activation, the output layer still uses the sigmoid activation.
- There wasn't much change with the results for this layer, but the accuracy was 0.002 better
- I kept 20 epochs to test the second layer.

Model 3:

After the slight increase in accuracy from the second model I decided to add a third layer and even more neurons.

```
268/268 - 0s - 2ms/step - accuracy: 0.7269 - loss: 0.5744
Loss: 0.5744125247001648, Accuracy: 0.7268804907798767
```

- This model had three input layers with 75, 50, and 25 neurons respectively.
- All three input layers used the relu activation while the output used sigmoid still
- After running this model, it was clear that more data cleaning may need to be done if the results are still about the same after three input layers and many more neurons.
- I upped the epoch layers to 100 to see if it would help, but it didn't seem to.

Model 4:

For the final model I did a little bit more data cleaning by dropping the SPECIAL_CONSIDERATIONS and the STATUS columns due the severe imbalance in hopes that it would help with the accuracy.

```
268/268 - 1s - 2ms/step - accuracy: 0.7277 - loss: 0.6384
Loss: 0.6384295225143433, Accuracy: 0.7276967763900757
```

- This model had four input layers with 100, 50, 25, and 15 neurons respectively.
- I ran this model with 500 epochs to see if it would help.
- The accuracy was still about the same as the other three models.
- There is still data preprocessing that need to be done to get the desired accuracy.

Summary

After all four of the models showed around the same accuracy it's clear that there needs to be more work with the data before working with the models. I was not able to hit the desired accuracy of 75%, the highest I was able to reach was 73%. After trying to increase the input layers, numbers of neurons per layer, and even changing some of the activations the results were about the same across all four models. This makes me think that there is something else that could be done to make the data better for the models.

In the future using a xgboost model after working more with the data could produce the desired accuracy results. This model uses several decision trees to improve the overall performance, and I think it would be beneficial to see what this model would look like. This model would also help with regularization to help minimize overfitting leading to better results.