Analysis report for Alphabet Soup Charity

Overview of the Analysis:

This project attempts to develop a tool using machine learning and neural networks algorithms which will be used to select the applicants for funding with the best chances of success in their ventures.
Bucketing of data, removal of non-essential columns, application of neural networks, training, transformation and fitting the data, determination of loss and accuracy was done with the provided dataset.

Initial Features provided in the dataset were :

- **EIN** and **NAME**—Identification columns
- **APPLICATION_TYPE**—Alphabet Soup application type
- **AFFILIATION**—Affiliated sector of industry
- **CLASSIFICATION**—Government organization classification
- **USE_CASE**—Use case for funding
- **ORGANIZATION**—Organization type
- **STATUS**—Active status
- **INCOME_AMT**—Income classification
- **SPECIAL_CONSIDERATIONS**—Special considerations for application
- **ASK_AMT**—Funding amount requested
- **IS_SUCCESSFUL**—Was the money used effectively

Data Processing :

- What variable(s) are the target(s) for your model?

  IS_SUCCESSFUL column was used as the target for the model.

- What variable(s) are the features for your model?

  NAME, APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, ASK_AMT columns were used for features.

- What variable(s) should be removed from the input data because they are neither targets nor features?

  Variable EIN was removed from the input data because it was neither target nor feature.

- **Compiling, Training, and Evaluating the Model**
  - How many neurons, layers, and activation functions did you select for your neural network model, and why?

    In my model, 3 hidden layers each with many neurons were used because the increased number of compute seemed to increase the accuracy. But increasing the

number of compute is expensive. For the first hidden layer activation **“relu”** was used and the other two hidden layers activation **“sigmoid”** was used. For output, activation **“sigmoid”** was used. Increasing number of hidden layers from two to three did not achieve the target accuracy level so data was readjusted by adding the **“NAME”** column which increased the accuracy to 81%.

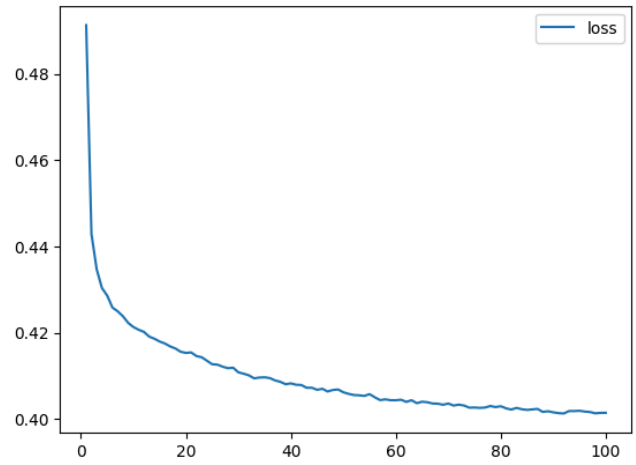
- Were you able to achieve the target model performance ?

  Yes. Maximum attained accuracy was 79.20%

```
268/268 - 1s - loss: 0.4470 - accuracy: 0.7902 - 549ms/epoch - 2ms/step
Loss: 0.4470379054546356, Accuracy: 0.7902041077613831
```

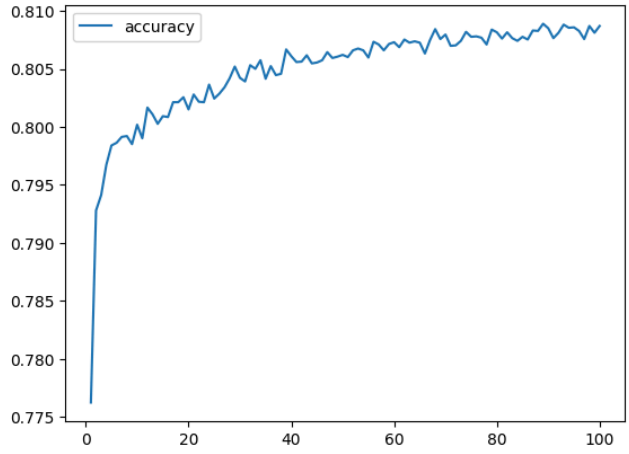
```
history_df.plot(y="loss")
```

```
<Axes: >
```




```
history_df.plot(y="accuracy")
```

```
<Axes: >
```



- What steps did you take in your attempts to increase model performance ?

  To attain the required accuracy, hidden layers were increased from two to three which did not give desired accuracy. Hence, "NAME" feature was added into the data which had the biggest impact on improving the efficiency and accuracy percentage.

3. **Summarize the overall results of the deep learning model. Include a recommendation for how a different model could solve this classification problem, and then explain your recommendation.**

Summary:

Overall the accuracy is above 75%. We are able to correctly classify each in the test data 75% of the time and applicant has a 80% chance of being successful if they follow this :
Name of applicants appear more than 5 times
Type of applicants following T3 T4 T5 T6 T7 T8 T10 T19
Application of following classification C1000, C1200, C2000, C2100, C3000

Recommendation :

A supervised machine learning can be a better way to classify the groups and result. Since the number of input parameters are higher, I think a random forest classifier may provide a reliable and accurate result. Performing deep learning algorithms using random forest classifiers might provide better accuracy.

```
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators = 120, random_state = 78)

rf_model = rf_model.fit(X_train_scaled, y_train)

y_prediction = rf_model.predict(X_test_scaled)

print(f"Random forest model accuracy {accuracy_score(y_test, y_prediction)}')
```