Analysis report for Alphabet Soup Charity

Overview of the Analysis:

This project attempts to develop a tool which helps to select the applicants for funding with the best chances of success in their ventures using machine learning and neural networks.

Bucketing of data, removal of non-essential columns, application of neural networks, training, transformation and fitting the data, determination of accuracy and loss was done.

Initial Features:

- **EIN and NAME**—Identification columns
- **APPLICATION_TYPE**—Alphabet Soup application type
- **AFFILIATION**—Affiliated sector of industry
- **CLASSIFICATION**—Government organisation classification
- **USE_CASE**—Use case for funding
- **ORGANIZATION**—Organisation 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 Preprocessing:

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

  IS_SUCCESSFUL column was used as target for the model.

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

  APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, ASK_AMT columns have been used for features.

- What variable(s) should be removed from the input data because they are neither targets nor features?
  NAME and **EIN** columns have been removed because they are not contributing to the result.

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, the number of neurons varied between 1 and 120, number of hidden layers varied between 1 and 5 layers, three activation functions namely relu, leaky relu and tanh functions were assessed for hidden layers and sigmoid activation function was used for output.

Based on the pre-processed data, it was understood that the number of input parameters are 43. Hence, the number of neurons in the hidden layers were varied upto 120.

Iterations were carried to estimate the loss and accuracy calculations. The following are the few graphs plotted during the iterations.

Case 1:

3 hidden layers

Number of input parameters: 43

Number of neurons in first hidden layer: 120

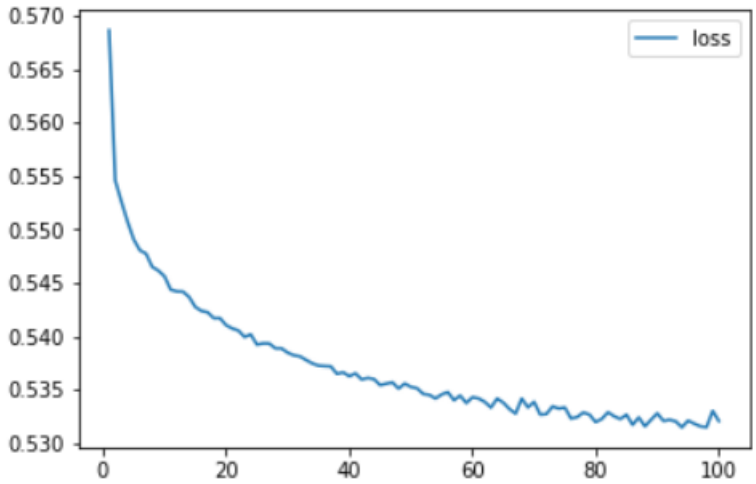
Number of neurons in second hidden layer: 90

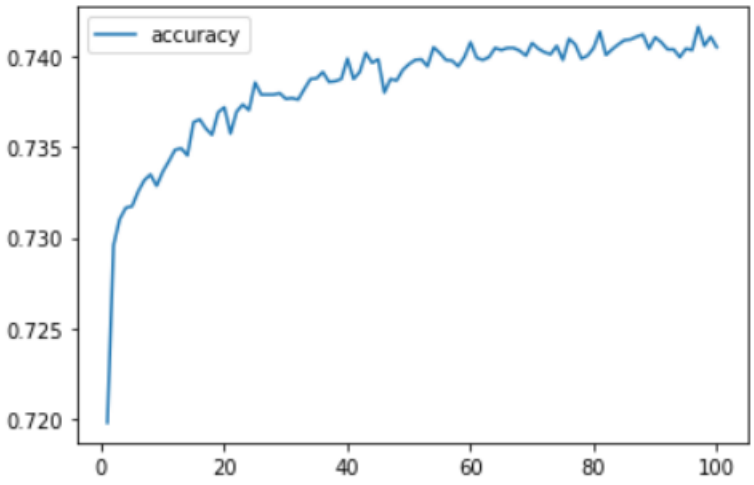
Number of neurons in third hidden layer: 60

Activation function used for hidden layers: Relu

Activation function used for output layer: Sigmoid

Number of epochs: 100





Case 2:

2 hidden layers

Number of input parameters: 43

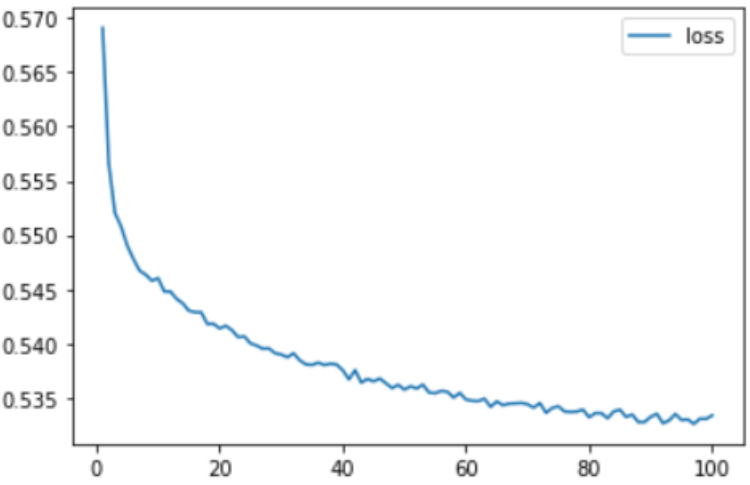
Number of neurons in first hidden layer: 120

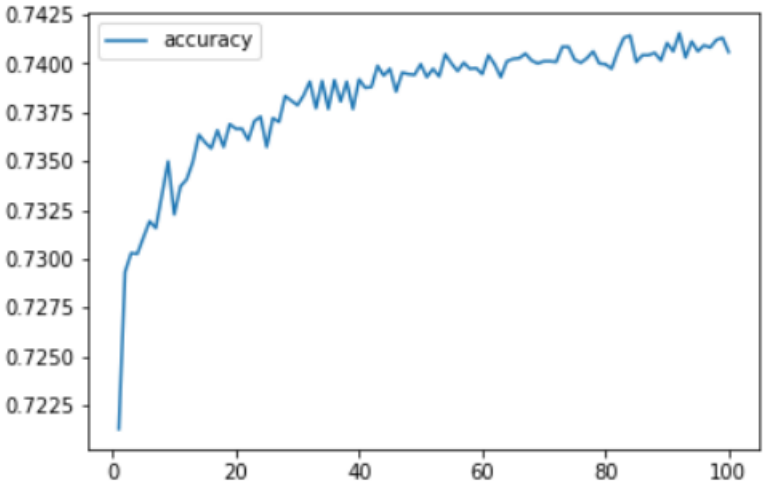
Number of neurons in second hidden layer: 90

Activation function used for hidden layers: Relu

Activation function used for output layer: Sigmoid

Number of epochs: 100





Case 3:

2 hidden layers

Number of input parameters: 43

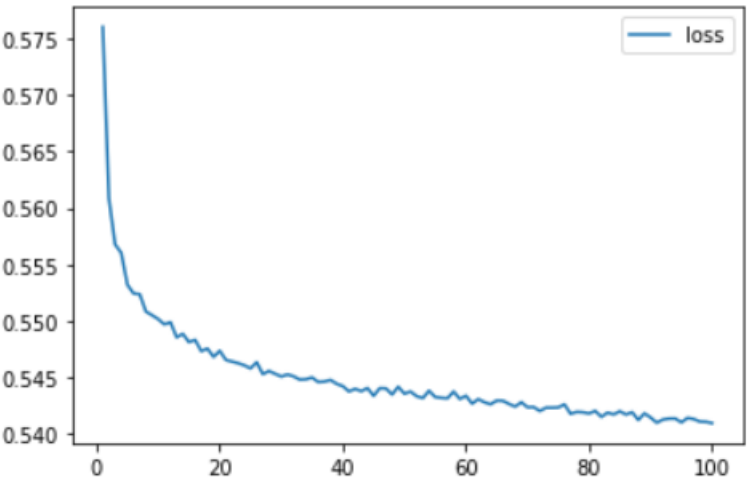
Number of neurons in first hidden layer: 120

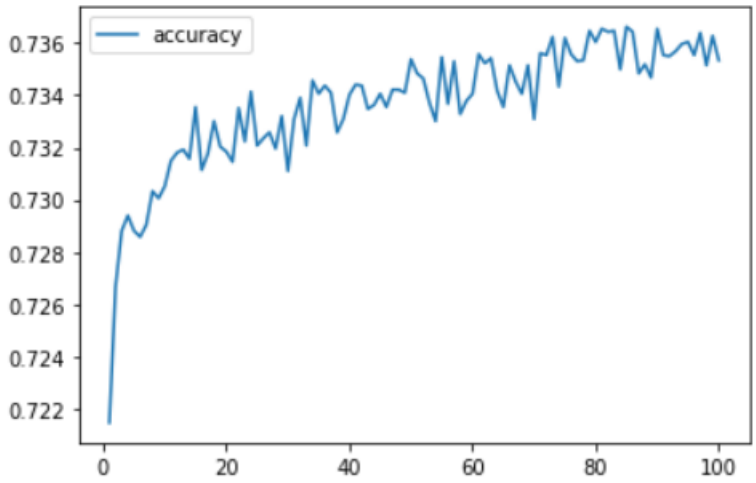
Number of neurons in second hidden layer: 90

Activation function used for hidden layers: LeakyRelu

Activation function used for output layer: Sigmoid

Number of epochs: 100





Accuracy calculation for the top three best Hyperparameters:

```
[20] # Evaluate the top 3 models against the test dataset
     top_model = tuner.get_best_models(3)
     for model in top_model:
         model_loss, model_accuracy = model.evaluate(X_test_scaled,y_test,verbose=2)
         print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 1s - loss: 0.5548 - accuracy: 0.7338 - 555ms/epoch - 2ms/step
Loss: 0.5547618269920349, Accuracy: 0.7337609529495239
268/268 - 1s - loss: 0.5531 - accuracy: 0.7336 - 586ms/epoch - 2ms/step
Loss: 0.5531494617462158, Accuracy: 0.7336443066596985
268/268 - 1s - loss: 0.5533 - accuracy: 0.7332 - 541ms/epoch - 2ms/step
Loss: 0.5532716512680054, Accuracy: 0.7331778407096863
```

- Were you able to achieve the target model performance?

  The maximum attained accuracy was 73.3%

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

  For attaining better accuracy, I varied the number of hidden layers between 1 and 5, I varied the number of neurons in the hidden layers between 1 and 120. I used tanh, relu and leakyrelu activation functions. I also test the model for 20 and 100 epochs. Overall, the best obtained accuracy was 0.73376.

Summary

Based on the graphs presented above, I think that the convergence of the solution has been attained indicating that changing the number of epochs may not play a crucial role. The smoothness of the curve (loss and accuracy) may be improved by changing the other parameters such as changing the number of neurons, changing the number of hidden layers and changing the activation functions. But the overall pattern indicates that the convergence of the solution has been attained and it may be very difficult to increase the accuracy further.

Recommendation:

A supervised machine learning can be 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 algorithm using random forest classifier might provide a better accuracy.