Performance of Charity funding predictor model

Overview [1]

The primary purpose of this analysis is that using the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup then optimize the predictor performance by optimization in dataset preprocess and model and show how much they affect the performance

Result

➢ Data preprocessing:

- **Target [2]** :
  - IS_SUCCESSFUL is considered the target for the model as the propose is whether applicants will be successful if funded.

- **Feature which was deleted [3]**
  - EIN and NAME were dropped and weren't considered being useful in first attempt
  - For optimizing EIN, STATUS, SPECIAL_CONSIDERATIONS were dropped as they are not correlated to target enough.
- **Features which were used [4]**
  - For the first attempt all the features were used except EIN, NAME**.**
  - But for optimizing the NAME column considered be useful so all features were used except EIN, STATUS, SPECIAL_CONSIDERATIONS .

➢ Compiling, Training, and Evaluating the Model:

- **Model before optimizing**: First a Sequential model with dense layers of Input layer, two hidden layers with 80 and 30 neurons and using a 'relu' activation function for both

---

[1] Overview of the analysis: Explain the purpose of this analysis.
[2] What variable(s) are considered the target(s) for your model?
[3] What variable(s) are neither targets nor features, and should be removed from the input data?
[4] What variable(s) are considered to be the features for your model?

hidden layers. One 'sigmoid' activation function with one neuron for output layer. (Figure 1). Model was fitted in only on training dataset (Figure 2)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 80)                3520
_________________________________________________________________
dense_1 (Dense)              (None, 30)                2430
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 31
=================================================================
Total params: 5,981
Trainable params: 5,981
Non-trainable params: 0
```

Figure 1: Model before optimizing

- **Model after optimizing:** a Sequential model with dense layers of Input layer, three hidden layers instead of two with 443 and 35, 3 neurons and using tanh activation function instead of relu for three hidden layers. One 'sigmoid' activation function with one neuron for output layer. As our input shape increased from 43 to 443, increasing the number of neurons in the first hidden layer improved the accuracy (Figure 3).[5]

```
input_shape=X_train_scaled.shape[1]
nn1 = tf.keras.models.Sequential()
#hidden layer
nn1.add(Dense(units=443, input_dim=input_shape, activation='tanh'))
nn1.add(Dense(35, activation='tanh'))
nn1.add(Dense(3, activation='tanh'))
# Output layer
nn1.add(Dense(1, activation='sigmoid'))

nn1.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 443)               196692
_________________________________________________________________
dense_23 (Dense)             (None, 35)                15540
_________________________________________________________________
dense_24 (Dense)             (None, 3)                 108
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 4
=================================================================
Total params: 212,344
Trainable params: 212,344
Non-trainable params: 0
```

Figure 2:Optimized model

[5] How many neurons, layers, and activation functions did you select for your neural network model, and why?

For achieving model performance firstly, I added neurons in first hidden as our input shape increases from 43 to 443, then I increased the neurons of second to 35 and then adding another hidden layer with 3 neurons. [6]

Summary:

before any optimization in data preprocessing and modeling (Figure 3):

- ✓ `Loss: 0.569678544998169, Accuracy: 0.7256559729576111`

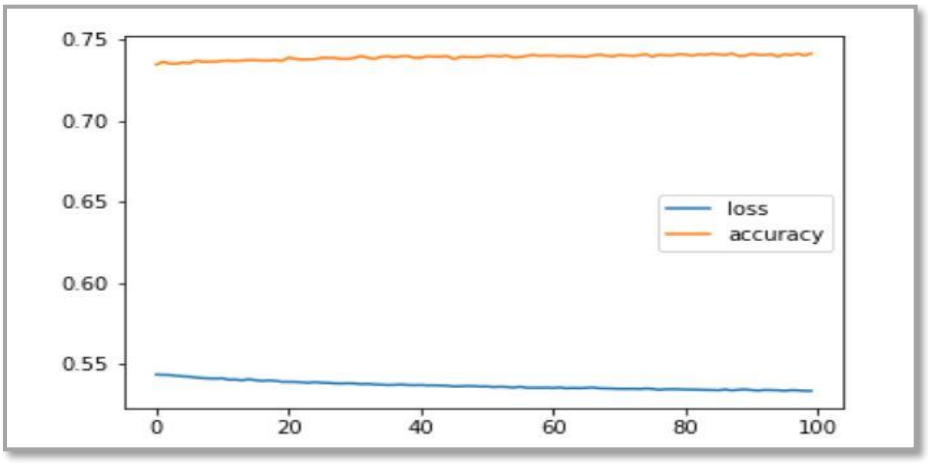

Figure 3 : Loss and Accuracy before any optimization

Step_1&2: In part of data preprocessing the NAME column wasn't considered as a noisy column and instead it categorized and binned.

In part of data preprocessing, The STATUS, SPECIAL_CONSIDERATIONS, and EIN columns were dropped as it doesn't be correlated to target much enough .

- ✓ `Loss: 0.4908713400363922, Accuracy: 0.7860058546066284`

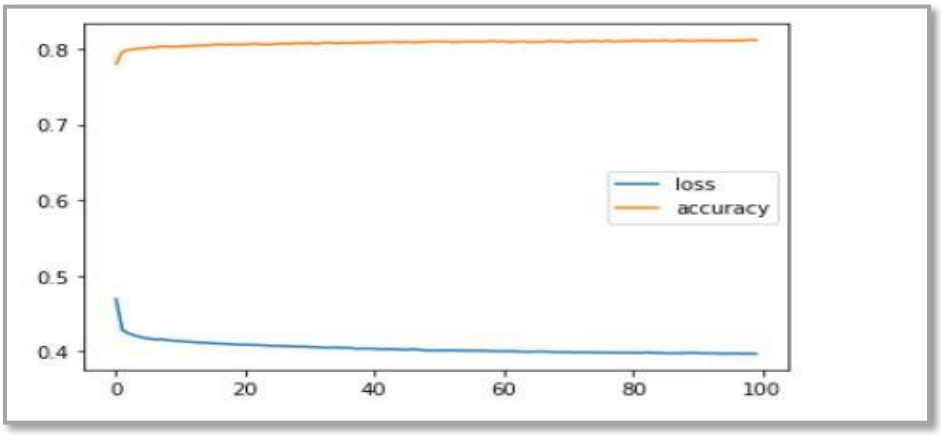

Figure 4: Loss and Accuracy after dataset preprocessing optimization

[6] What steps did you take to try and increase model performance?

Step_3: In part of model making: Additional neurons are added to the hidden layers. Additional hidden layers are added. The activation function of hidden layers are changed from relu to tanh. Adding early stopping callback for decreasing overfitting:

1. Changing 80 neurons to 443：

   ✓ `Loss: 0.452499121427536, Accuracy: 0.7845481038093567`

2. Changing 30 neurons to 35：

   ✓ `Loss: 0.45344942808151245, Accuracy: 0.7868804931640625`

3. Adding a hidden layer by 3 neurons before output layer:

   ✓ `Loss: 0.4491606652736664, Accuracy: 0.7879008650779724`

So, we achieved the target model performance, our Loss was decease from 0.57 to 0.45 and accuracy was increased from 0.72 to .79 before and after optimization.[7]

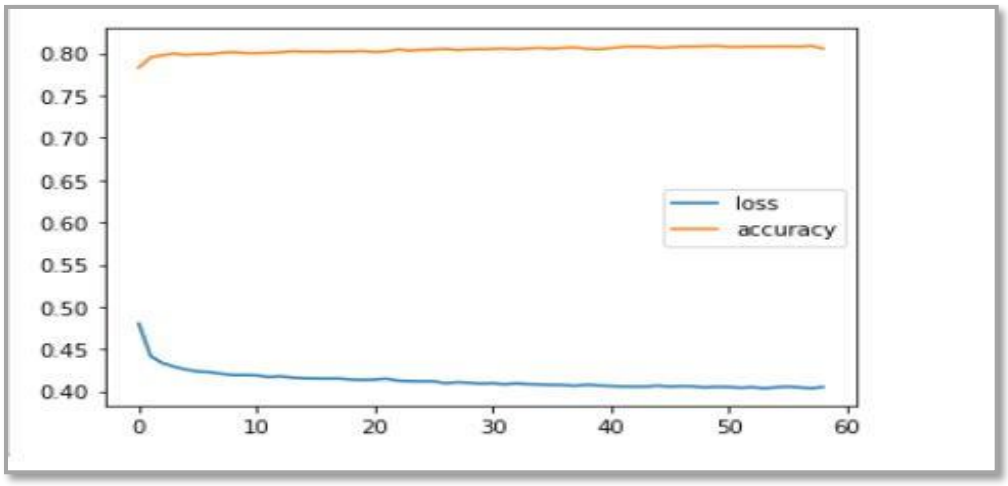

Figure 5:loss and accuracy after optimization

[7] Were you able to achieve the target model performance?