Report on the Neural Network Model

Overview of the Analysis: This analysis aims to build a deep learning model using TensorFlow and Keras to predict which applicants are most likely to succeed if funded by Alphabet Soup, a nonprofit organization. The objective is to develop a predictive tool based on historical data from successful and unsuccessful funding applications, enabling the identification of high-potential applicants and improving resource allocation efficiency.

Results

1. **Data Preprocessing:**

   - What variable(s) are the target(s) for your model?
     - IS_SUCCESSFULL column was used as the target variable to indicate whether the funding application was successful (1) or unsuccessful (0).
   - What variable(s) are the features for your model?
     - All other columns were used as the features of the model.

     ```
     NAME
     APPLICATION_TYPE
     AFFILIATION
     CLASSIFICATION
     USE_CASE
     ORGANIZATION
     STATUS
     INCOME_AMT
     SPECIAL_CONSIDERATIONS
     ASK_AMT
     ```

   - What variable(s) should be removed from the input data because they are neither targets nor features?
     - For the optimized model I just removed the (EIN) column which was a unique identifier that was not needed for this model

2. Compiling, Training, and Evaluating the Model

- How many neurons, layers, and activation functions did you select for your neural network model, and why?
  - Input Layer:
    Neurons: 100
    Activation Function: relu
  - Hidden Layers (additional two layers):
    Neurons: 135, 128
    Activation Function: relu
  - Output Layer:
    Neurons: 1
    Activation Function: Sigmoid

  The selection of neurons and layers was driven by the observed enhancement in prediction accuracy, allowing the model to more effectively capture complex patterns.

Layer (type)	Output Shape	Param #
dense_28 (Dense)	(None, 100)	7,200
dense_29 (Dense)	(None, 135)	13,635
dense_30 (Dense)	(None, 128)	17,408
dense_31 (Dense)	(None, 1)	129

Total params: 38,372 (149.89 KB)

Trainable params: 38,372 (149.89 KB)

Non-trainable params: 0 (0.00 B)

- Were you able to achieve the target model performance?
  Yes, the model's accuracy was 75.16% with the loss of 52.52%.

- What steps did you take in your attempts to increase model performance?
    - Featured Engineering phase: I added the NAME column as it seems one of the most relevant feature maybe because of containing meaningful distinctions between entities (like different types of organizations, companies, or projects) as one of the categorical columns which significantly improved my model's performance. It contained 19568 unique values; I combined the values less than 100 in a new column named "Other". Also binned the AFFILIATION column values less than 100, binning other categorical columns such as USE_CASE, INCOME_AMT didn't affect the performance improvement thus I removed them.
    - Model Infrastructure: Changes I made to the initial infrastructure model are:
      In the input layer → the number of units were increased from 80 to 100
      Hidden Layers → added one more hidden layer so we got two layers, one with 135 units and the other with 128 units.

      Epochs → were revised from 100 to 50 to decrease the training time and prevent from overfitting (preventing the model from learning noise and irrelevant details)

3. Summary:

After training, the model achieved an **accuracy score of 75.16%** on the validation/test dataset, which met the target of 75%. Despite various optimization attempts, including altering activation functions, adjusting network architecture, and changing optimizers, improvement in performance was minimal, with accuracy staying between 74% and 75%. While the model met the desired accuracy, experimenting with additional hyperparameters such as learning rate schedules or different optimizers (e.g., RMSprop or Nadam) could improve performance further. Additionally, more feature engineering could help identify important relationships.