Deep Learning Challenge

Isaiah Rollie

Analysis Overview

The purpose of this analysis is to develop a deep learning model to predict whether an organization applying for funding from Alphabet Soup will be successful. By training a neural network on past application data, we aim to classify future applicants accurately, thereby optimizing the funding decision-making process.

Data Preprocessing

- Target Variable: The target variable for this model is `IS_SUCCESSFUL`, which indicates whether an applicant received funding (1) or not (0).
- Feature Variables: The features used in the model include `APPLICATION_TYPE`, `AFFILIATION`, `CLASSIFICATION`, `USE_CASE`, `ORGANIZATION`, `STATUS`, `INCOME_AMT`, `SPECIAL_CONSIDERATIONS`, and `ASK_AMT`. These variables provide key insights into the nature of the organization and its application status.
- Removed Variables: `EIN` and `NAME` were removed as they are unique identifiers and do not contribute to the predictive capability of the model.

Compiling, Training, and Evaluating the Model

- Neurons, Layers, and Activation Functions:
    - The initial model consisted of three hidden layers. The number of neurons in each layer was determined using a general rule of thumb based on the number of input features.
    - The `ReLU` activation function was used in the hidden layers to introduce non-linearity, while the `sigmoid` activation function was used in the output layer to predict binary classifications.

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 9
hidden_nodes_layer2 = 18

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="relu"))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="relu"))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```

- Model Performance: The initial model achieved an accuracy of 72%, which was below the target benchmark of 75%

Layer (type)	Output Shape	Param #
dense_15 (Dense)	(None, 9)	378
dense_16 (Dense)	(None, 18)	180
dense_17 (Dense)	(None, 1)	19

```
268/268 - 0s - 2ms/step - accuracy: 0.7257 - loss: 0.5585
Loss: 0.5584793090820312, Accuracy: 0.7257142663002014
```

- Optimization Steps Taken:
    - The dataset was refined by reintroducing the `NAME` variable to determine its impact on model performance.
    - Additional neurons were added to each layer to allow the network to capture more complex relationships.
    - The number of epochs was increased to provide the model with more training opportunities.
    - Different activation functions and dropout layers were tested to prevent overfitting.
    - These improvements resulted in an optimized model accuracy of 78%, exceeding the initial goal.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	3,171
dense_1 (Dense)	(None, 14)	112
dense_2 (Dense)	(None, 1)	15

```
268/268 - 0s - 1ms/step - accuracy: 0.7860 - loss: 0.4639
Loss: 0.4638599157333374, Accuracy: 0.7860058546066284
```

Summary

The deep learning model demonstrated its ability to classify successful and unsuccessful funding applications with an accuracy of 79%. While this model showed improvement after optimization, other machine learning models such as Random Forest or Gradient Boosting could be explored for comparison. These models may provide better interpretability and require less fine-tuning compared to deep learning models. In cases where computational efficiency and

feature importance insights are needed, a tree-based model could offer a viable alternative to a neural network for this classification problem.