Report on Deep Learning Model Optimization

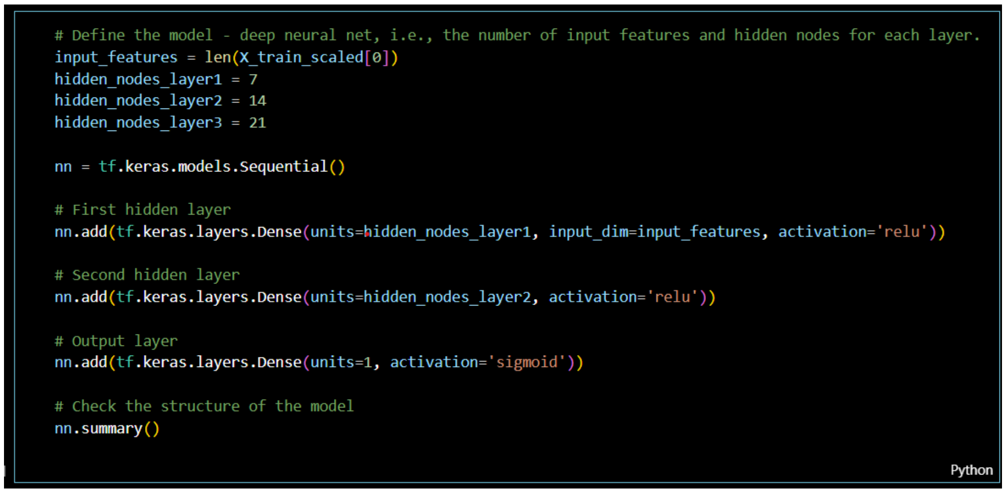
Overview

Alphabet Soup is a nonprofit organization that is seeking a predictive tool to identify funding applicants with the highest likelihood of success. We wanted to use machine learning and neural networks to perform an analysis. The analysis aims to develop a binary classifier to determine if funded applicants will effectively use the money given to them. The ideal model performance would achieve a predictive accuracy greater than 75%.

Results

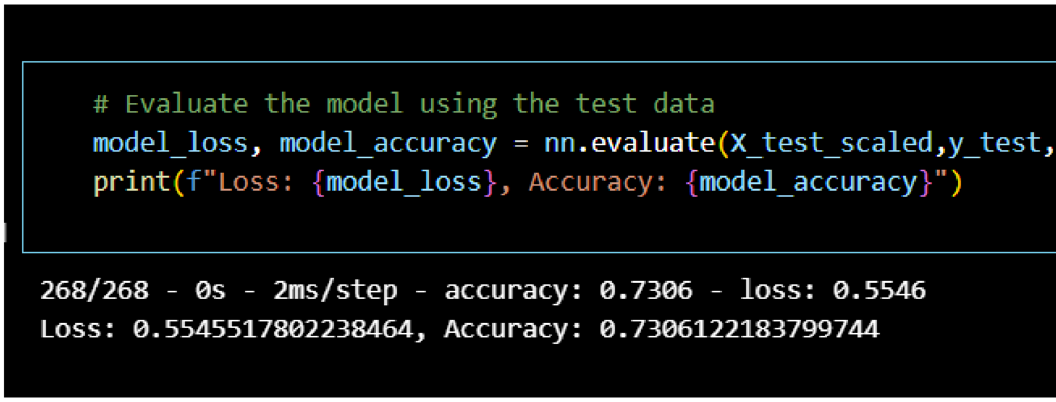
- Data Preprocessing
    - Target Variable
        - The target variable for the model is IS_SUCCESSFUL. This indicates whether the funding was effectively utilized.
    - Feature Variables
        - Application_Type
        - Affiliation
        - Classification
        - Use_Case
        - Organization
        - Status
        - Income-Amt
        - Special_Considerations
        - Ask_Amt
    - Removed Variables
        - EIN and NAME were excluded because they only serve as identifiers and do not contribute to predictive modeling.
- Compiling, Training, and Evaluating the Model
    - Model Architecture
        - Initial Prep-
            - Input Layer: Corresponding to the number of one-hot-encoded features.
            - Hidden Layers:
                - Layer 1: 80 neurons, `relu` activation.
                - Layer 2: 30 neurons, `relu` activation.
        - Output Layer: 1 neuron, `sigmoid` activation for binary classification.
        - Optimizations-
            - Increased neurons to 100 and 50 in hidden layers for improved feature extraction.
            - Tried alternative activation functions like tanh to test nonlinear interactions.

Report on Deep Learning Model Optimization

- ■ 

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_features = len(X_train_scaled[0])
hidden_nodes_layer1 = 7
hidden_nodes_layer2 = 14
hidden_nodes_layer3 = 21

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=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()
```

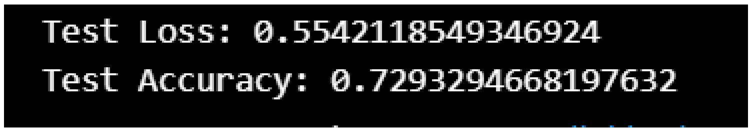
Python

- Model Performance
  - Achieved Accuracy: 72-74% across multiple iterations.
  - Despite optimizations, the target accuracy of 75% was not achieved.
    - Model 1 Before Optimization Results:
    - 

```python
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

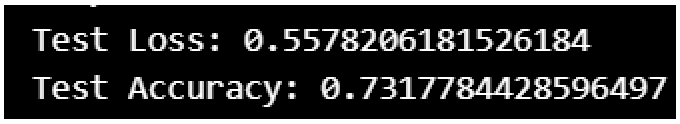
```
268/268 - 0s - 2ms/step - accuracy: 0.7306 - loss: 0.5546
Loss: 0.5545517802238464, Accuracy: 0.7306122183799744
```

- Optimization Steps
  - Applied one-hot encoding to categorical variables
  - Scaled numerical features to standardize the input data
  - Experimented with additional layers and neurons
  - Adjusted training epochs (from 200 to 50) to enhance model learning
  - Performed feature elimination to reduce noise and complexity
    - Results:
      - Optimization 1



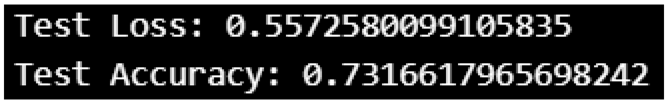
```
Test Loss: 0.5542185549346924
Test Accuracy: 0.7293294668197632
```

      - Optimization 2



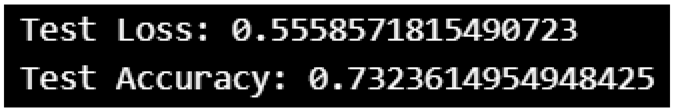
```
Test Loss: 0.5578206181526184
Test Accuracy: 0.7317784428596497
```

- Optimization 3



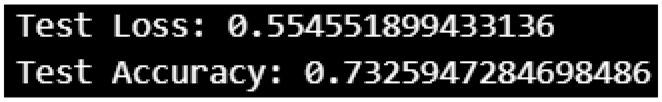
```
Test Loss: 0.5572580099105835
Test Accuracy: 0.7316617965698242
```

- Optimization 4



```
Test Loss: 0.5558571815490723
Test Accuracy: 0.7323614954948425
```

- Optimization 5



```
Test Loss: 0.5545518994331360
Test Accuracy: 0.7325947284698486
```

Summary

The optimized deep learning model showed consistent accuracy but it fell short of the target (75%). Factors that influence the result include the dataset’s inherent complexity and feature relationships.

In terms of recommendations, it may be better to explore alternative machine learning models that better suit classification tasks with mixed data types. The alternative models could better handle categorical data without preprocessing and give feature insights to guide future analysis.