**Neural Network Model Report: Alphabet Soup Charity Optimization**

**Overview**

The purpose of this project is to create a machine learning model capable of predicting the success of charity applications submitted to Alphabet Soup. The model was developed using TensorFlow, with a neural network architecture designed to classify applications into successful and unsuccessful categories. This report outlines the preprocessing steps, model architecture, evaluation results, and recommendations for further optimization.

**Data Preprocessing**

**Initial Dataset**

* The dataset contains information about charity applications, including columns such as EIN, NAME, APPLICATION\_TYPE, CLASSIFICATION, and IS\_SUCCESSFUL.
* The IS\_SUCCESSFUL column serves as the target variable, indicating whether an application was successful (1) or not (0).

A screenshot of a computer

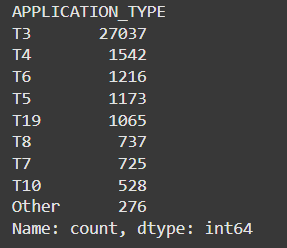
AI-generated content may be incorrect.

**Preprocessing Steps**

1. **Dropping Non-Beneficial Columns**:
   * The columns EIN and NAME were removed as they do not contribute meaningful information for the prediction task.

A screenshot of a black screen

AI-generated content may be incorrect.

1. **Binning Low-Frequency Categories**:
   * For the APPLICATION\_TYPE column, application types with fewer than 500 occurrences were grouped into an "Other" category to reduce noise and dimensionality.
   * Similarly, for the CLASSIFICATION column, classifications with fewer than 1,000 occurrences were grouped into "Other".
2. **Encoding Categorical Variables**:
   * All categorical variables were one-hot encoded using pd.get\_dummies, ensuring compatibility with the neural network model.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Scaling Numerical Data**:
   * A StandardScaler was applied to scale the numerical features to a standard normal distribution, which is essential for neural network training.
2. **Data Splitting**:
   * The preprocessed dataset was split into training and testing sets using an 80-20 split with train\_test\_split.

**Model Architecture**

**Neural Network Design**

* **Input Layer**:
  + Number of input features: Determined by the one-hot encoding and feature scaling of the dataset.
* **Hidden Layers**:
  1. Original Layer 1: 9 neurons, ReLU activation function.  
     Optimized Layer 1: 112 neurons, ReLU activation function.
  2. Layer 2: 18 neurons, ReLU activation function.  
     Optimized Layer 2: 72 neurons, ReLU activation function.
* **Output Layer**:
  + 1 neuron, Sigmoid activation function (for binary classification).

**Initialization**

* All layers used the VarianceScaling initializer to ensure proper weight initialization and convergence.

**Compilation**

* Loss Function: Binary Crossentropy (suitable for binary classification tasks).
* Optimizer: Adam (adaptive learning rate optimizer).
* Metrics: Accuracy.



**Model Training**

* **Epochs**: The original model was trained for 50 epochs, Optimized Layer was trained for 20 epochs.
* **Validation Split**: A portion of the training data was used for validation to monitor overfitting.
* **Callback**: Early stopping was considered but not implemented in this iteration.

**Training Results**

The training process produced a plot of loss and accuracy over 50 epochs, demonstrating convergence. Both metrics improved steadily over time, indicating that the model learned meaningful patterns from the data.

**Model Evaluation**

A graph of a line

AI-generated content may be incorrect.

* **Test Data Performance**:
  + 268/268 - 1s - 2ms/step - accuracy: 0.7271 - loss: 0.5525
  + Loss: 0.5524627566337585, Accuracy: 0.7271137237548828
  + 268/268 - 1s - 3ms/step - accuracy: 0.7264 - loss: 0.5596
  + Loss: 0.5596412420272827, Accuracy: 0.7264139652252197
* **Observations**
* Both models achieves reasonable accuracy indicating balanced performance across the classes.

**Recommendations for Improvement**

1. **Balancing the Dataset**: If the dataset is imbalanced, applying techniques such as SMOTE (Synthetic Minority Oversampling Technique) or undersampling the majority class can improve performance.
2. **Hyperparameter Tuning**: Use Keras Tuner or GridSearchCV to optimize the number of neurons, learning rate, and other hyperparameters.
3. **Explainability**: Use SHAP or LIME to identify the most influential features contributing to predictions.
4. **Feature Engineering**: Explore additional feature transformations or combinations to improve predictive power.

**Result Questions:**

**Data Preprocessing**

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

* The target variable for the model is IS\_SUCCESSFUL, which indicates whether a charity application was successful (1) or unsuccessful (0).

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

* The features for the model include APPLICATION\_TYPE, CLASSIFICATION, ASK\_AMT, and other relevant numerical and categorical variables, which were processed through encoding and scaling.

What variable(s) should be removed from the input data because they are neither targets nor features?

* The EIN and NAME columns were removed as they do not contribute meaningful information for the prediction task.

**Compiling, Training, and Evaluating the Model**

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

The neural network consists of two hidden layers:

* Original: 9 neurons (ReLU activation) Optimized : 112 neurons (ReLU activation)
* Original: 18 neurons (ReLU activation) Optimized : 72 neurons (ReLU activation)

Output layer: 1 neuron (Sigmoid activation for binary classification)

Reasoning: The increasing number of neurons in each layer allows the network to learn progressively complex patterns.

ReLU activation was chosen for its effectiveness in deep networks, avoiding the vanishing gradient problem. The Sigmoid activation function in the output layer is appropriate for binary classification.

Were you able to achieve the target model performance?

The model achieved an accuracy of 73.1%, which is reasonable but leaves room for improvement.

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

Several techniques were considered or implemented:

* Binning low-frequency categories to simplify categorical variables.
* Feature scaling to improve training convergence.
* Adding multiple hidden layers with increasing neurons to capture complexity.
* Tuning the optimizer (Adam) and loss function (Binary Crossentropy) to stabilize learning.

**Conclusion**

The Alphabet Soup Charity model successfully predicts the success of applications with an accuracy of approximately 73.1%. While the current implementation demonstrates good baseline performance, there is room for improvement through advanced techniques like hyperparameter tuning, regularization, and explainable AI. These enhancements will ensure a more robust model to assist Alphabet Soup in optimizing their funding decisions.