Overview of the Analysis

The goal of this analysis was to build a binary classification model to predict whether an organization funded by Alphabet Soup is likely to be successful. Using a deep learning neural network, we leveraged historical funding data to train a model that classifies organizations as either successful (1) or unsuccessful (0).

The analysis included data preprocessing, model compilation, training, and optimization, with multiple attempts to improve performance.

Results

Data Preprocessing

Target Variable:

- IS_SUCCESSFUL → Indicates whether the organization was successful.

Feature Variables:

- APPLICATION_TYPE
- CLASSIFICATION
- INCOME_AMT
- ASK_AMT
- SPECIAL_CONSIDERATIONS (before optimization)
- ORGANIZATION

Removed Variables:

- EIN and NAME (Identification numbers, not useful for prediction)
- AFFILIATION, USE_CASE, SPECIAL_CONSIDERATIONS, STATUS: Removed after optimization for being unnecessary or redundant.

Compiling, Training, and Evaluating the Model

Initial Model (Baseline)

- **Structure:**
    - **2 Hidden Layers**
    - **First Layer: 80 neurons, ReLU activation**
    - **Second Layer: 30 neurons, ReLU activation**
    - **Output Layer: Sigmoid activation**
- **Training Setup:**
    - **Optimizer: Adam**

    - **Loss Function: Binary Crossentropy**
    - **Epochs: 100**
- **Results:**
    - **Final Accuracy: ~72.3%**
    - **Loss: 0.5578**

Optimization Attempts

First Optimization

- **Changes:**
    - Increased **neurons** in each layer (256 → 128)
    - Added an **extra hidden layer**
    - Used **LeakyReLU** instead of ReLU in one layer
    - Switched from **StandardScaler to MinMaxScaler**
- **Results:**
    - **Accuracy: 73.0%** (Slight improvement)

Second Optimization

- **Changes:**
    - Increased neurons further (512 → 256 → 128 → 64)
    - Used **Swish activation** in one hidden layer
    - Switched from **Adam to Nadam optimizer**
    - Implemented **learning rate reduction callback**
- **Results:**
    - **Accuracy: 73.0%** (No significant improvement)

Third Optimization

- **Changes:**
    - Used **RobustScaler** instead of MinMaxScaler
    - Tried **SELU activation** in one layer
    - Switched **Loss function to Hinge Loss**
    - Added **Class Weights** to handle imbalances in the dataset
- **Results:**

- **Accuracy dropped to 57.3%**
- **Significant loss increase (~164104)**

Summary

Overall Model Performance

- **Baseline Model: 72.3% Accuracy**
- **Optimized Models:** Highest achieved accuracy was **73.0%**
- **Final Attempt:** Accuracy dropped to **57.3%**

Key Observations

- Adding **more layers and neurons** did not significantly improve accuracy.
- Changing **activation functions** (LeakyReLU, Swish, SELU) had minimal impact.
- The **Adam optimizer** performed better than Nadam in this dataset.
- Using **MinMaxScaler** was more effective than RobustScaler.
- **Class Weights** did not improve performance but instead reduced accuracy.

Recommendation

Since deep learning did not significantly improve performance beyond 73%, an alternative approach such as:

1. **Random Forest Classifier** (for better interpretability)
2. **XGBoost** (for improved accuracy with structured data)
3. **Hyperparameter Tuning with Grid Search** for neural networks

could be more effective in improving predictive performance.