Alphabet Soup Deep Learning Model Report

1. Overview of the Analysis

The purpose of this analysis is to build a binary classification model for Alphabet Soup, a nonprofit foundation. The goal is to predict whether an applicant will successfully use the funding provided by Alphabet Soup. This model can assist the organization in identifying which applicants have a higher likelihood of success based on historical data. Using a neural network, we aimed to develop a predictive model with an accuracy higher than 75%.

2. Results

2.1 Data Preprocessing

Target Variable:

* The target variable for this analysis is `IS\_SUCCESSFUL`, a binary indicator of whether the funding was used successfully (1 = successful, 0 = unsuccessful).

Feature Variables:

* The features used for the model include:
  + `APPLICATION\_TYPE`: The type of application submitted.
  + `AFFILIATION`: The sector of industry the applicant is associated with.
  + `CLASSIFICATION`: The government classification of the applicant organization.
  + `USE\_CASE`: The intended purpose for the funding.
  + `ORGANIZATION`: The type of organization applying for the funding.
  + `INCOME\_AMT`: The income classification of the organization.
  + `SPECIAL\_CONSIDERATIONS`: Whether the application has special considerations (e.g., veterans, disability).

Removed Variables:

* `EIN` and `NAME` were removed from the dataset because these are identification columns that do not add any predictive value to the model.

2.2 Compiling, Training, and Evaluating the Model

Neurons, Layers, and Activation Functions:

* Number of Layers: The model was built with two hidden layers for simplicity, to avoid overfitting due to an overly complex model.
* Neurons per Layer:
  + The first hidden layer contained 64 neurons.
  + The second hidden layer contained 32 neurons.

Activation Function:

* `tanh` was selected for both hidden layers. This activation function was chosen for its ability to output negative values, which sometimes improves performance on smaller datasets.
* `sigmoid` was used in the output layer for binary classification.

Model Performance:

* Accuracy: After training and evaluating the model, the highest accuracy achieved was 73%, which is below the target accuracy of 75%.

Steps Taken to Improve Performance:

* SMOTE: To address class imbalance in the `IS\_SUCCESSFUL` target variable, SMOTE was applied to oversample the minority class and balance the dataset.
* Simplification of the Model: Initially, a more complex architecture was attempted with three hidden layers, but this was reduced to two hidden layers, which helped improve the model’s generalization.
* Hyperparameter Tuning: Various learning rates, dropout rates, and regularization techniques were applied. After tuning, a learning rate of `0.001` and a dropout rate of 20% were chosen to avoid overfitting.

2.3 Model Optimization

Model 1:

* **Feature Selection (Variance Threshold):**
  + The model begins by removing lowvariance features from the dataset. Using Variance. Threshold(threshold=0.01), it eliminates features that do not change much (those with little variance). These features are considered less informative for the model's prediction, improving model efficiency by focusing on more significant features.
* **Feature Scaling**:
  + The selected features are scaled using StandardScaler. This ensures that each feature has a mean of 0 and a standard deviation of 1, which helps the neural network converge faster by preventing dominance of features with larger ranges.
* **Building a Deep Neural Network:**
* The model is built as a **deep neural network** with four hidden layers:
  + **First Hidden Layer**: Contains 256 neurons, **BatchNormalization** to normalize outputs, **LeakyReLU** activation to allow small negative gradients, and **Dropout (30%)** to reduce overfitting.
  + **Second Hidden Layer**: Contains 128 neurons with similar regularization (BatchNormalization, LeakyReLU, Dropout).
  + **Third Hidden Layer**: Contains 64 neurons with similar regularization.
  + **Fourth Hidden Layer**: Contains 32 neurons with similar regularization.
* **L2 Regularization** is applied to all layers to prevent overfitting by penalizing large weights.
* **Output Layer**:
  + The output layer contains **1 neuron** with a **sigmoid activation function**, used for **binary classification** (0 or 1).
* **Model Compilation**:
  + The model is compiled using **RMSprop optimizer** with a small learning rate (0.0005), which ensures slow and steady learning. The loss function is **binary crossentropy**, which is suitable for binary classification tasks. The metric tracked is **accuracy**.
* **Model Training**:
  + The model is trained with **early stopping** to prevent overfitting. Early stopping monitors the validation loss (val\_loss) and halts training if it does not improve for 40 consecutive epochs, while restoring the best weights encountered during training. The model is trained for a maximum of 200 epochs with a batch size of 64.

Model 2:

* Data Preprocessing:
  + The dataset is loaded, and irrelevant columns (`EIN`, `NAME`, `STATUS`, `SPECIAL\_CONSIDERATIONS`, and `ASK\_AMT`) are removed.
  + Rare categories in the `APPLICATION\_TYPE` and `CLASSIFICATION` columns are binned into an "Other" category to reduce noise.
  + Categorical variables are converted into numeric form using onehot encoding.
  + The data is split into training and testing sets, and features are standardized using `StandardScaler`.
* Neural Network Architecture:
* Three hidden layers:
* First Layer: 128 neurons, L2 regularization (`0.0005`), ReLU activation, and 20% dropout.
* Second Layer: 64 neurons, L2 regularization, ReLU activation, and 20% dropout.
* Third Layer: 32 neurons, L2 regularization, ReLU activation, and 20% dropout.
* Output Layer: A single neuron with a sigmoid activation function for binary classification (predicts `IS\_SUCCESSFUL`).
* Model Compilation:
* The model is compiled using the Adam optimizer with a learning rate of `0.001` and binary crossentropy as the loss function.
* Training:
* The model is trained for up to 150 epochs with early stopping (patience of 20 epochs) to prevent overfitting. The best model weights are restored when validation loss stops improving.
* Evaluation:
* The model is evaluated on the test data, achieving an optimized accuracy and loss, and then saved as `AlphabetSoupCharity\_Simplified.h5`.

This model simplifies the deep learning architecture while still incorporating techniques like regularization, dropout, and early stopping to prevent overfitting and improve generalization.

Model 3:

### Concise Summary of the Model

* Data Preprocessing:
* Columns Dropped: Irrelevant columns (`EIN`, `NAME`, `STATUS`, `SPECIAL\_CONSIDERATIONS`, `ASK\_AMT`) are removed.
* Binning Rare Categories: Rare values in `APPLICATION\_TYPE` and `CLASSIFICATION` are grouped into an "Other" category to reduce noise and improve model generalization.
* OneHot Encoding: Categorical variables are converted into numerical format using onehot encoding.
* Train/Test Split: Data is split into training and testing sets (80/20 ratio).
* Scaling: The features are standardized using `StandardScaler` to ensure that all features have a mean of 0 and standard deviation of 1.
* SMOTE for Class Balancing: SMOTE (Synthetic Minority Oversampling Technique) is applied to balance the classes in the target variable (`IS\_SUCCESSFUL`), which helps to prevent bias toward the majority class.
* Neural Network Architecture:
  + Two hidden layers:
  + First Layer: 64 neurons with tanh activation, L2 regularization (`0.0001`), and 20% dropout to prevent overfitting.
  + Second Layer: 32 neurons with tanh activation, L2 regularization, and 20% dropout.
  + Output Layer: A single neuron with sigmoid activation for binary classification.
* Model Compilation:
  + The model is compiled using the Adam optimizer with a learning rate of `0.001`. Binary crossentropy is used as the loss function, and the accuracy metric is tracked.
* Training:
  + The model is trained for a maximum of 150 epochs with early stopping. Early stopping is triggered if the validation loss does not improve for 20 consecutive epochs. This ensures that the best model is restored to avoid overfitting.
* Evaluation:
  + After training, the model is evaluated on the test dataset. The optimized loss and accuracy are printed, and the model is saved as `AlphabetSoupCharity\_Simplified\_v2.h5`.

This model uses a simpler neural network architecture with class balancing (SMOTE) and early stopping to improve generalization and prevent overfitting.

**3. Summary of Results**

Model Accuracy: Despite tuning and applying SMOTE to balance the classes, the model reached a maximum accuracy of 73%. This is below the 75% target accuracy.

Challenges:

The dataset's structure may not have been complex enough for a neural network to perform optimally.

Neural networks tend to perform better on larger datasets or datasets with unstructured data like text or images.

**4. Recommendation for an Alternative Model**

To potentially improve performance and reach the target accuracy of over 75%, it is recommended to explore traditional machine learning algorithms like XGBoost or Random Forest. These models often perform better on structured/tabular data like the dataset provided by Alphabet Soup.

**5. Why Use XGBoost or Random Forest?**

Better suited for tabular data: Unlike neural networks, which excel at unstructured data, algorithms like XGBoost and Random Forest handle structured data with categorical and numerical features more efficiently.

Feature Importance: Treebased models like Random Forest and XGBoost can automatically capture feature importance and interactions between variables, which is difficult for a neural network unless feature engineering is extensive.

Generalization: Treebased algorithms tend to generalize better with fewer tuning requirements compared to neural networks, especially when data size is moderate.

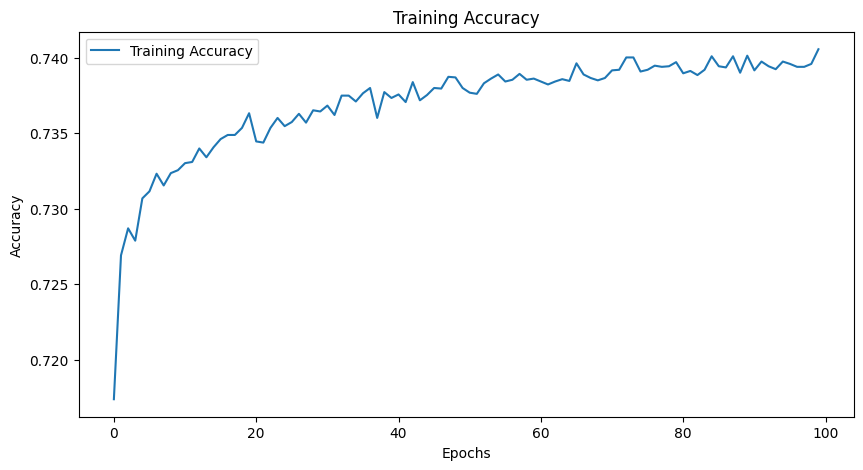
**6. Potential Steps for Improvement:**

* Feature Selection: Use Recursive Feature Elimination (RFE) or other techniques to identify and remove less important features before applying XGBoost or Random Forest.
* Ensemble Methods: Combine multiple models (e.g., Neural Network + Random Forest) using ensemble techniques like Voting Classifier to improve overall accuracy.
* Crossvalidation: Use crossvalidation to get a more reliable estimate of the model’s performance.

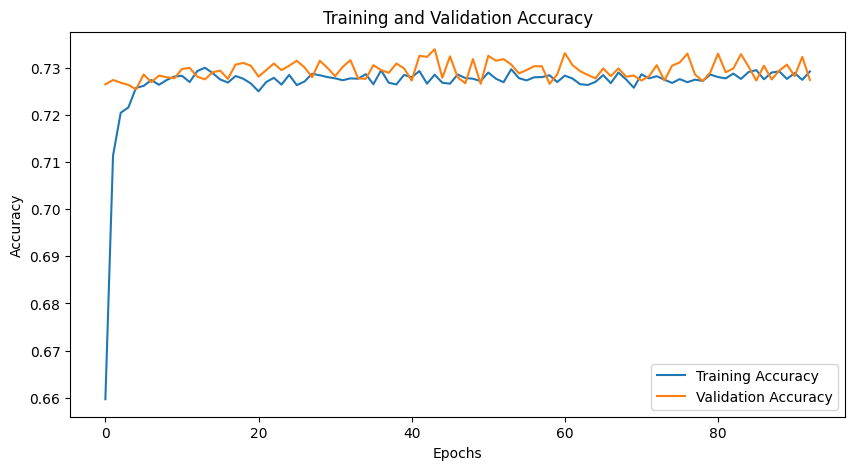
**7. Images for Training and Validation Performance:**

To visualize the model's training and validation performance, we can examine the following learning curves.

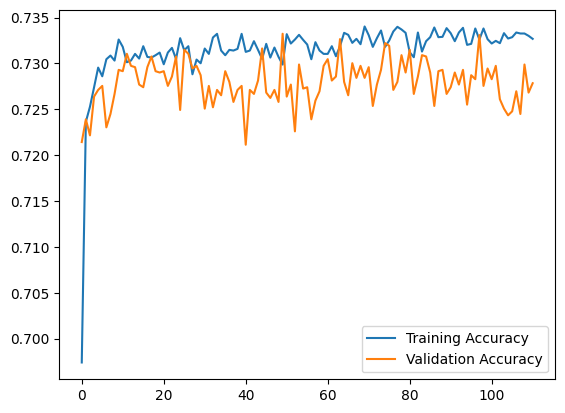
Original Model



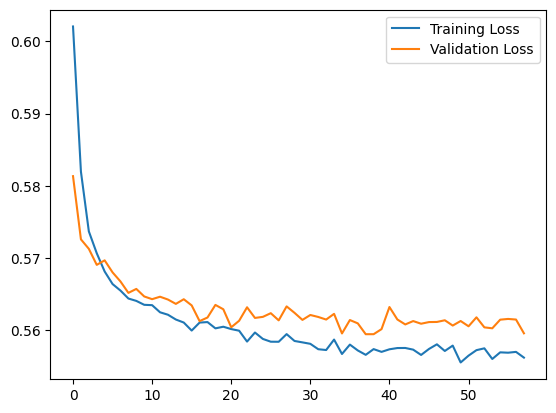
Model 1



Model 2



Model 3



**8. Conclusion**

In conclusion, the deep learning model applied to the Alphabet Soup dataset achieved an accuracy of 73% after extensive tuning, but failed to meet the 75% target. Given the nature of the dataset, traditional machine learning models like XGBoost or Random Forest might perform better, and these models should be explored to improve accuracy and generalization.