NEURAL NETWORK FOR ALPHABET SOUP CHARITY

Report - *Module 21 challenge – Data Analytics Bootcamp*

Key words: Neural Network, Deep Learning, Keras, TensorFlow, Scikit-learn

Summary

This report describes the coding for a neural network model using Keras, TensorFlow, and Scikit-learn libraries to predict whether Alphabet Soup Charity applicants will be successful if funded. The report outlines the data pre-processing steps, including handling missing values, grouping categories, and scaling numerical features. It also discusses the design of the neural network architecture, including the use of dropout regularization. The report also includes the evaluation of the model's performance, including accuracy and loss metrics, and provides suggestions for further improvements.

Objective

The purpose of this analysis is to develop a deep learning model using the Alphabet Soup Charity dataset (containing data from various organizations that have received funding from Alphabet Soup). The model aims to predict which organizations are likely to be successful after receiving funding from Alphabet Soup, based on the features provided in the dataset.

Method

Modelling workflow:

1. Decide on a model and create a model instance.
2. Split the dataset into training and testing sets and process the data.
3. Train/fit the training data to the model.
4. Evaluate the model for predictions and transformations.

Results

The target variable for the model is **IS_SUCCESSFUL**.

Pre-processing was done as follows :

(refer to "***1. DeepLearning.ipynb***")

- Drop useless ID columns: The EIN and NAME columns are do not provide any useful information for the model. Dropping them reduces the data dimension and thus simplify the feature space.

- Replace low-frequency APPLICATION_TYPE categories: to reduce the number of unique categories in APPLICATION_TYPE and replace categories with fewer instances with "Others" to prevent overfitting and to generalize the model better.
- Combine low-frequency AFFILIATION categories: To simplify the feature space and reduce dimensionality, the low-frequency AFFILIATION categories are grouped under "Other".
- Replace low-frequency CLASSIFICATION categories: In order to prevent overfitting and improve model generalization, the low-frequency CLASSIFICATION categories are replaced with "Other".
- Combine low-frequency USE_CASE categories: The low-frequency USE_CASE categories are grouped as "Other" to simplify the feature space and reduce dimensionality.
- Combine low-frequency ORGANIZATION categories: The low-frequency ORGANIZATION categories are grouped under "Other" to simplify the feature space and reduce dimensionality.
- Remove STATUS and SPECIAL_CONSIDERATIONS columns: The STATUS and SPECIAL_CONSIDERATIONS columns are removed as they do not provide useful information for the model, simplifying the feature space.
- Convert INCOME_AMT column to a categorical variable and use one-hot encoding: To convert non-numeric data into numeric data that the model can work with, the INCOME_AMT column is converted to a categorical variable and one-hot encoded.
- Fill missing INCOME_AMT values with mode: To avoid bias in the model, the missing INCOME_AMT values are replaced with the mode value to fill gaps in the data.
- Scale the ASK_AMT feature: MinMaxScaler is used to normalize the range of the ASK_AMT feature and bring it to the same scale as the other features, preventing any single feature from dominating the model.
- Convert categorical data to numeric using pd.get_dummies: AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, and APPLICATION_TYPE categorical data are converted into numeric data using pd.get_dummies, creating binary variables for each category. This allows the model to work with categorical data and include it in the analysis.

(refer to "***2. Visualisations.ipynb***")

Figure 1 shows the top 10 features in the Alphabet Soup Charity dataset that are most strongly correlated with the target variable is_successful. The correlations are calculated using the Pearson correlation coefficient, which ranges from -1 to 1, with higher absolute values indicating stronger correlations. The chart shows that features such as APPLICATION_TYPE_T19, APPLICATION_TYPE_T3,

and INCOME_0 have the strongest positive correlations with is_successful, while features such as APPLICATION_TYPE_Other and AFFILIATION_CompanySponsored have the strongest negative correlations. The chart provides insights into which features may be the most important predictors of success for organizations receiving funding from Alphabet Soup Charity.

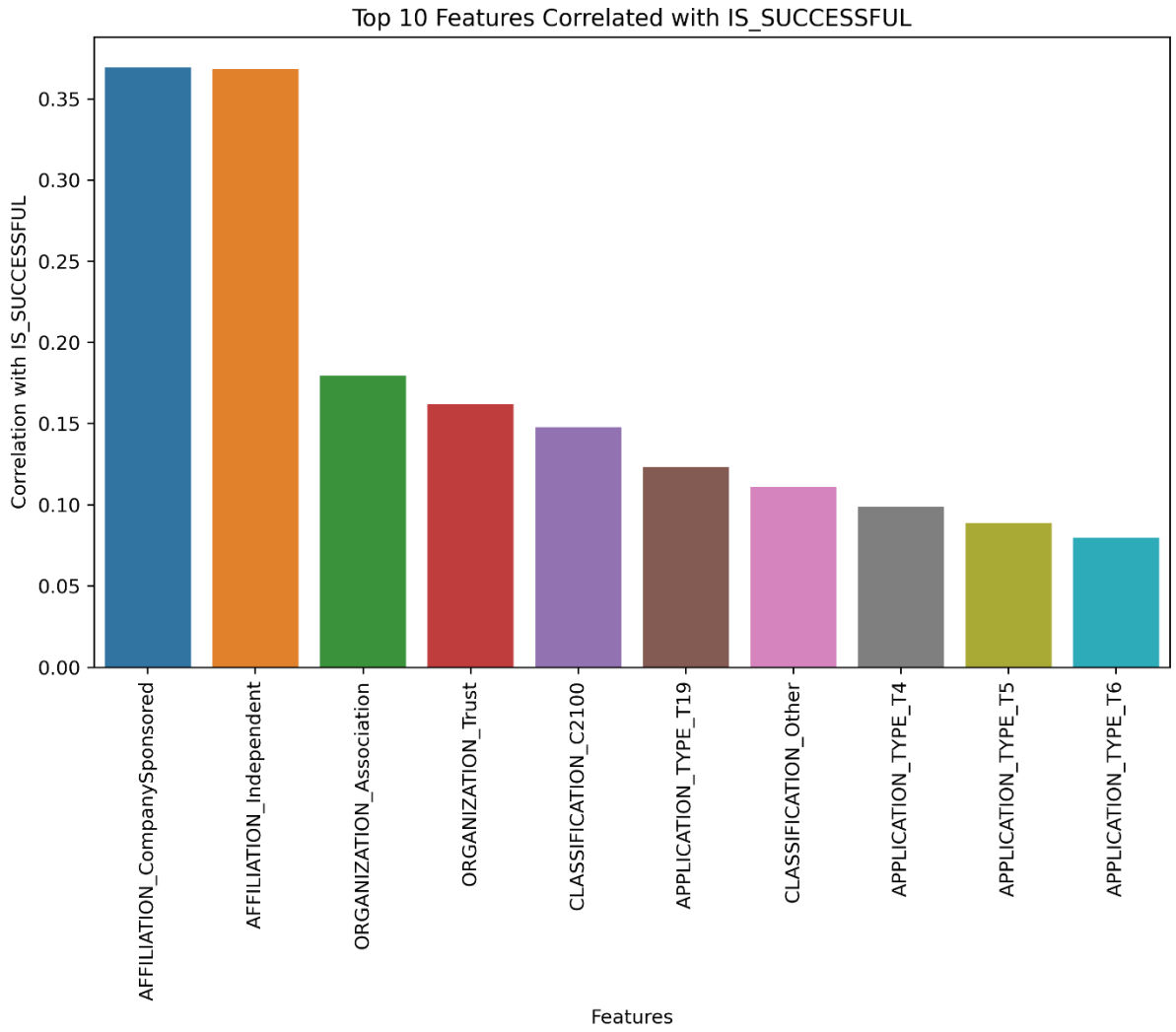

Figure 1 - Top 10 Features Correlated with the Target Value IS_SUCCESSFUL

Compiling, Training, and Evaluating the Model

The created neural network model consists of 1 input layer, 3 hidden layers with varying numbers of neurons, and 1 output layer. The activation functions used were "tanh" and "sigmoid". To automatically select the best hyperparameters for the model, Kerastuner was also used. (refer to *"3.* ***AlphabetSoupCharity_Optimisation.ipynb***")

During testing, an accuracy of 0.7283 was achieved on the test data, which is close to the target model performance of 0.75. To improve the model performance, various methods were attempted, such as

adding more hidden layers, increasing the number of neurons in each layer, changing the activation functions, and tuning the optimizer's learning rate.

Figure 2 presents the training and validation loss and accuracy of the neural network model throughout the training epochs. The blue and orange lines correspond to the training and validation loss, respectively, while the green and red lines represent the training and validation accuracy, respectively. The graph can be utilized to identify overfitting or underfitting in the model and to determine the optimal number of epochs to train the model.

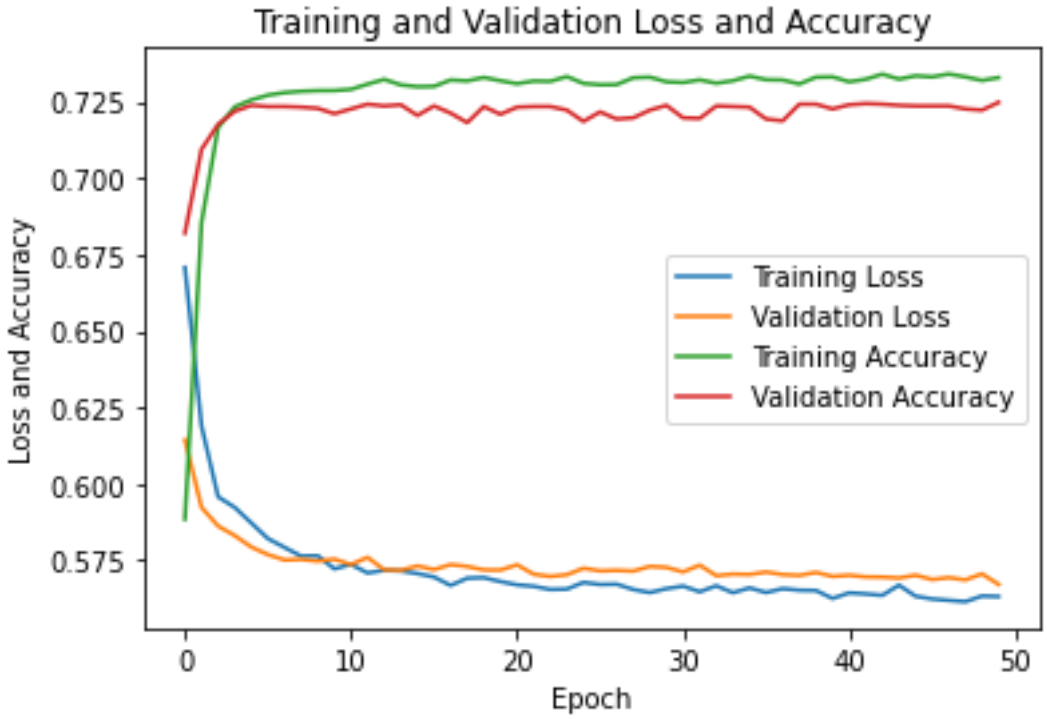

Figure 2 - Training and Validation Loss and Accuracy

Summary

In summary, this deep learning model performed reasonably well in predicting whether an organization will be successful or not. However, maybe a different model, such as a random forest classifier or a support vector machine, should be explored to see if it can achieve better performance on this classification problem. These models are known to work well with binary classification problems and may be more suitable for the data provider