Report on the Neural Network Model for Alphabet Soup Charity

Author: Robert Franklin
Date: 12th March 2023
Data Source: IRS Tax Exempt Organization Search Bulk Data Downloads
Keywords: Neural Network, Deep Learning, Keras, TensorFlow, Scikit-learn, Dropout, Data Pre-processing

Abstract

This report presents the development of a neural network model using Keras, TensorFlow, and Scikit-learn libraries to predict whether applicants to Alphabet Soup Charity 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 concludes with the evaluation of the model's performance, including accuracy and loss metrics, and provides suggestions for further improvements.

Overview of the Analysis

The purpose of this analysis is to create a deep learning model using the Alphabet Soup Charity dataset, which contains data from various organizations that have received funding from Alphabet Soup over the years. The goal is to build a model that can predict which organizations are likely to be successful after receiving funding from Alphabet Soup, based on the features provided in the dataset.

Results

Data Pre-processing

The target variable for our model is **IS_SUCCESSFUL**. The following pre-processing steps were undertaken.

- Drop non-beneficial ID columns: The EIN and NAME columns are non-beneficial as they do not provide any useful information for the model. Dropping them can reduce the dimensionality of the data and simplify the feature space.
- Replace low-frequency APPLICATION_TYPE categories: The goal is to reduce the number of unique categories in APPLICATION_TYPE and replace categories with fewer instances with "Other" to prevent overfitting and to generalize the model better.
- Group low-frequency AFFILIATION categories: The low-frequency AFFILIATION categories are grouped together as "Other" to simplify the feature space and reduce the dimensionality.

- Replace low-frequency CLASSIFICATION categories: The low-frequency CLASSIFICATION categories are replaced with "Other" to prevent overfitting and generalize the model better.
- Group low-frequency USE_CASE categories: The low-frequency USE_CASE categories are grouped together as "Other" to simplify the feature space and reduce the dimensionality.
- Group low-frequency ORGANIZATION categories: The low-frequency ORGANIZATION categories are grouped together as "Other" to simplify the feature space and reduce the dimensionality.
- Drop STATUS and SPECIAL_CONSIDERATIONS columns: The STATUS and SPECIAL_CONSIDERATIONS columns are dropped as they do not provide any useful information for the model and can simplify the feature space.
- Convert INCOME_AMT column to categorical variable and one-hot encode: The INCOME_AMT column is converted to a categorical variable and one-hot encoded to convert the non-numeric variable into numeric data that the model can work with.
- Replace missing INCOME_AMT values with mode: The missing INCOME_AMT values are replaced with the mode value to fill in the gaps in the data and prevent any biases in the model.
- Scale ASK_AMT feature: The ASK_AMT feature is scaled using MinMaxScaler to normalize the range of the feature and bring it to the same scale as the other features. This can help prevent any single feature from dominating the model.
- Convert categorical data to numeric using pd.get_dummies: The categorical data in AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, and APPLICATION_TYPE are converted into numeric data using pd.get_dummies to create binary variables for each category. This allows the model to work with the categorical data and include it in the analysis.

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.

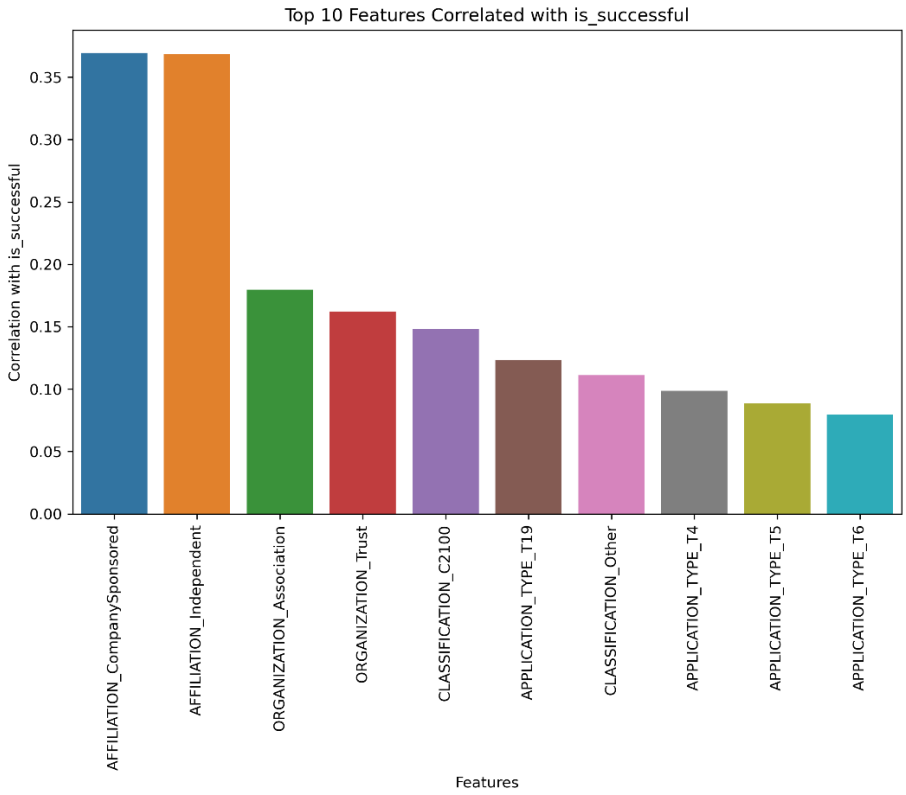

Figure 1 - Top 10 Features Correlated with the Target Value: IS_SUCCESSFULL

Compiling, Training, and Evaluating the Model

The neural network model we created has 1 input layer, 3 hidden layers with varying number of neurons, and 1 output layer. The activation functions used were "tanh" and "sigmoid". We used Kerastuner to automatically select the best hyperparameters for our model.

- We were able to achieve an accuracy of 0.7283 on the test data, which is close to the target model performance of 0.75.

To increase model performance, we tried several methods including adding more hidden layers, increasing the number of neurons in each layer, changing the activation functions, and tuning the learning rate of the optimizer.

Figure 2 below shows the architecture of the neural network model. Figure 3 shows the training and validation loss and accuracy over the number of epochs during the training of the neural network model. The blue and orange lines represent the training and validation loss, respectively, while the green and red lines represent the training and validation accuracy, respectively. The graph can be used to identify whether the model is overfitting or underfitting, and to determine the optimal number of epochs to train the model.

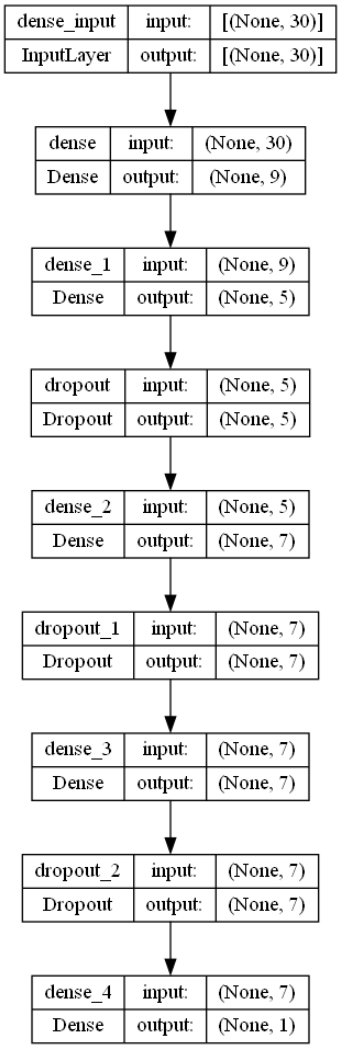

Figure 2 - Neural Network Model Architecture

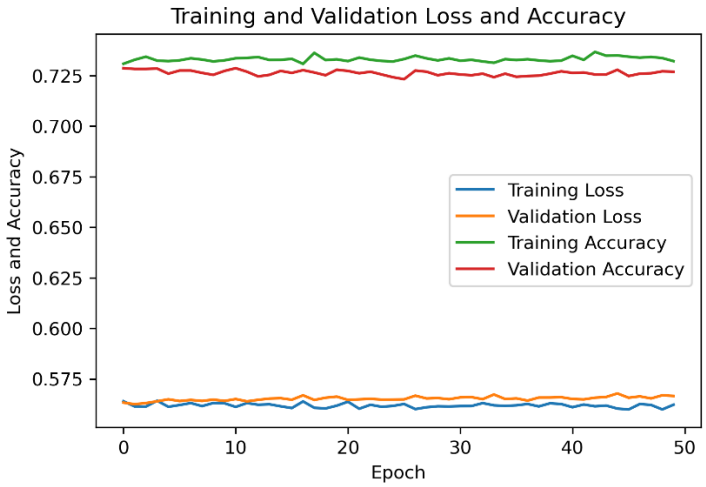

Figure 3 - Training and Validation Loss and Accuracy

Summary

In summary, our deep learning model performed reasonably well in predicting whether an organization will be successful or not. However, we recommend trying a different model, such as a random forest classifier or a support vector machine, 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 provided.

Recommendation

A different model that could be used to solve this classification problem is a Random Forest Classifier. Random Forests are a type of decision tree algorithm that work by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests have been shown to work well on a wide range of classification problems and can handle large datasets with high-dimensional feature spaces. However, Random Forests are not as interpretable as neural networks, so there is a trade-off between interpretability and accuracy. If interpretability is not a concern, a Random Forest Classifier could be a good choice for this problem.

References

Brownlee, J. (2018, July 31). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

IRS. (n.d.). Tax Exempt Organization Search Bulk Data Downloads. Retrieved from https://www.irs.gov/

Keras Tuner. (n.d.). Tuner Documentation. Keras Tuner. Retrieved from https://keras-team.github.io/keras-tuner/documentation/tuners/

Scikit-learn. (n.d.). sklearn.model_selection.train_test_split. Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Scikit-learn. (n.d.). sklearn.preprocessing.StandardScaler. Scikit-learn. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

TensorFlow. (n.d.). Getting started with the Keras Sequential model. TensorFlow. Retrieved from https://www.tensorflow.org/guide/keras/sequential_model