Deep Learning

Charity Funding Predictor

Ramon Faylona

Background

The non-profit foundation Alphabet Soup wants to create an algorithm to predict whether applicants for funding will be successful. Applying machine learning and neural networks on the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

From Alphabet Soup's business team, a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. Within this dataset are metadata about each organization, such as the following:

- **EIN** and **NAME**—Identification columns
- **APPLICATION_TYPE**—Alphabet Soup application type
- **AFFILIATION**—Affiliated sector of industry
- **CLASSIFICATION**—Government organization classification
- **USE_CASE**—Use case for funding
- **ORGANIZATION**—Organization type
- **STATUS**—Active status
- **INCOME_AMT**—Income classification
- **SPECIAL_CONSIDERATIONS**—Special consideration for application
- **ASK_AMT**—Funding amount requested
- **IS_SUCCESSFUL**—Was the money used effectively

Data Processing

Initial modelling of the data, EIN and NAME was removed from the model. The remaining columns were considered as features for the model. CLASSIFICATION and APPLICATION_TYPE were replaced with 'Other' due to high fluctuation. The dataset was then split into training and testing sets of data. The target variable of the model is 'IS_SUCCESSFUL' as identified by the value 1, which is a considered true and 0 was false. The application data was then analyzed, using CLASSIFICATION as the value used for binning. Unique values uses several data points as a cutoff point to bin 'rare' categorical values together in a new value, 'Other'. Categorical values were encoded by 'pd.get_dummies()'.

Compiling, Training, and Evaluating the Model

Neural Network was used on the model with two layers initially.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 80)                3520

 dense_1 (Dense)              (None, 30)                2430

 dense_2 (Dense)              (None, 1)                 31

=================================================================
Total params: 5,981
Trainable params: 5,981
Non-trainable params: 0
_________________________________________________________________
```

It generated 5,981 parameters and achieved a result of 73% accuracy, which was well short of the 75% accuracy required.

Optimization

On the optimization attempt reintroduced 'NAME' back into the dataset and used it to bin values before performing the rest of the model. Also in the model, a third hidden node and layer were added to help with the computation of the data.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                35760

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 20)                620

 dense_3 (Dense)             (None, 1)                 21

=================================================================
Total params: 38,831
Trainable params: 38,831
Non-trainable params: 0
_________________________________________________________________
```

By doing these the model was able to achieve an accuracy of 79%, 4% high than the required target.

Conclusion

In the attempts I made in hyper tuning the model, introducing more data, and adding additional layers helped achieve the accuracy required. Hyper tuning the model especially, binary natured problems, increasing the data and computational aspect of the AI seemed to help improve accuracy of the model.