Module 21 - Neural Networks and Deep Learning

# Background

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures.

With machine learning and neural networks, this analysis aims to use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

# Data Preprocessing

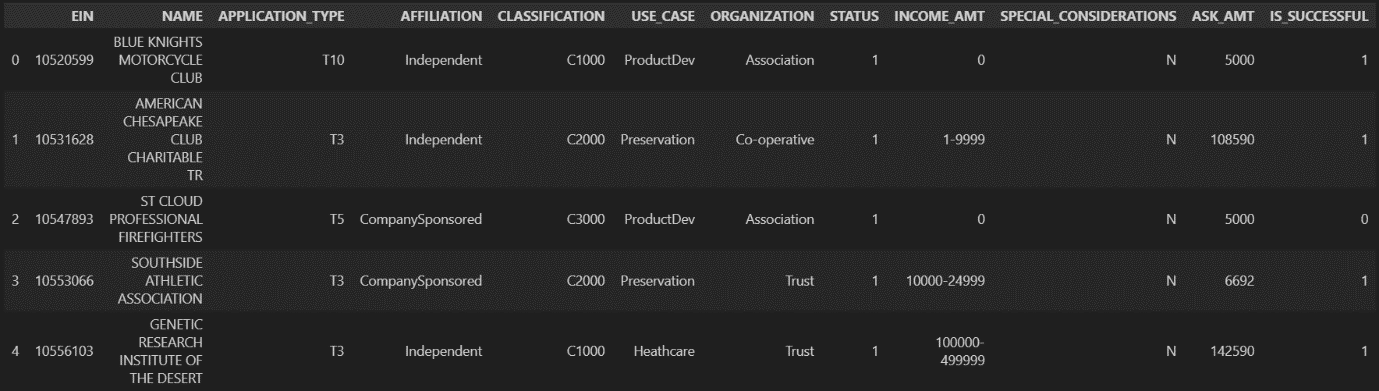
## Input data

The charity\_data csv data has the following columns, from which a DataFrame is created

|  |  |
| --- | --- |
| **Column Name** | **Description** |
| EIN | Identification column |
| NAME | Identification column |
| 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 considerations for application |
| ASK\_AMT | Funding amount requested |
| IS\_SUCCESSFUL | Was the money used effectively |

## Features and Target

Features



REMOVE

Target

*Figure 1 – sorting the input data into targets and features*

As shown in figure 1, the following columns are deleted from the DataFrame as they are neither target nor feature:

* EIN
* NAME

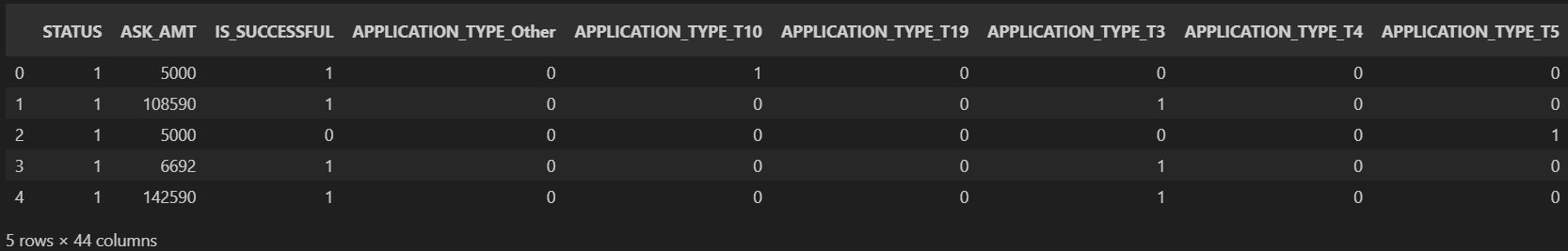
The following columns are identified as features:

* APPLICATION\_TYPE
* AFFILIATION
* CLASSIFICATION
* USE\_CASE
* ORGANIZATION
* STATUS
* INCOME\_AMT
* SPECIAL\_CONSIDERATIONS
* ASK\_AMT

The following column is identified as a target:

* IS\_SUCCESSFUL

## Categorical Data



*Figure 2 – Partial view of the DataFrame after categorical variables have been converted.*

The following columns contain categorical data, so they are converted into dummy variables with pd.get\_dummies(), as shown in figure 2:

* APPLICATION\_TYPE
* AFFILIATION
* CLASSIFICATION
* USE\_CASE
* ORGANIZATION
* INCOME\_AMT
* SPECIAL\_CONSIDERATIONS

For columns with more than 10 unique, values, any rare and unique values were merged together into the same categories, in order to reduce the total number of features. This applied to the following columns:

* APPLICATION\_TYPE
  + “T9”, “T13”, “T12”, “T2”, “T25”, “T14”, “T29”, “T15”, and “T17” were combined into 1 bin called “Other”.
* CLASSIFICATION
  + All categories EXCEPT for “C1000”, “C2000”, “C1200”, “C3000”, “C2100” were combined into 1 bin called “Other”.

## Training/testing split

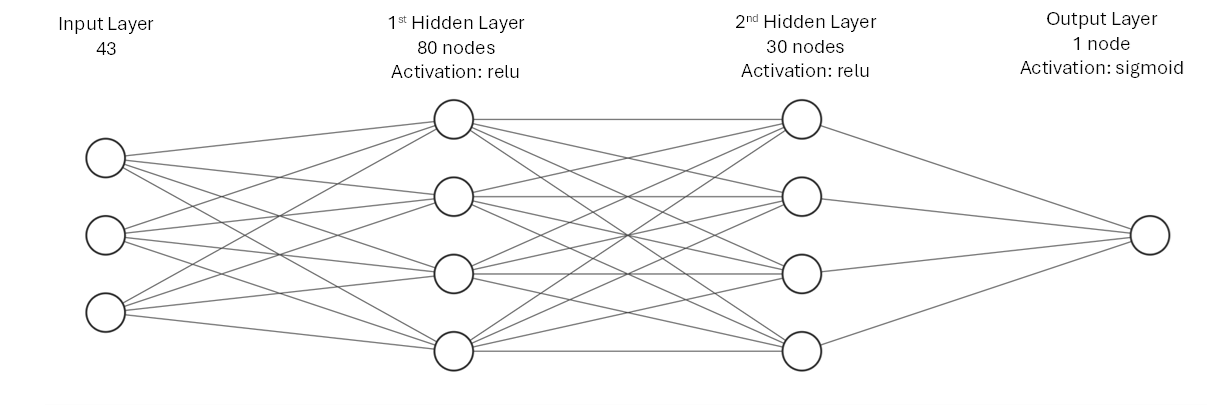
The data is split into training/testing features and target arrays, with train\_test\_split. The random state is set to 1 (for future reproducibility).

## Scaling

The training and testing features datasets are scaled with StandardScaler(), and are ready to be complied and fitted into the neural network.

# Compiling, Training, and Evaluating the Model

## Compiling



*Figure 3 – Stylised diagram of the neural network*

The neural network is structured in the following way, as shown in figure 3, with 2 hidden layers and 1 output layer.

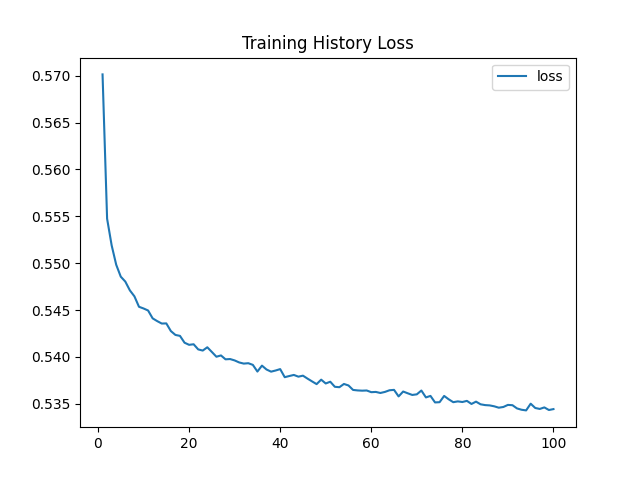
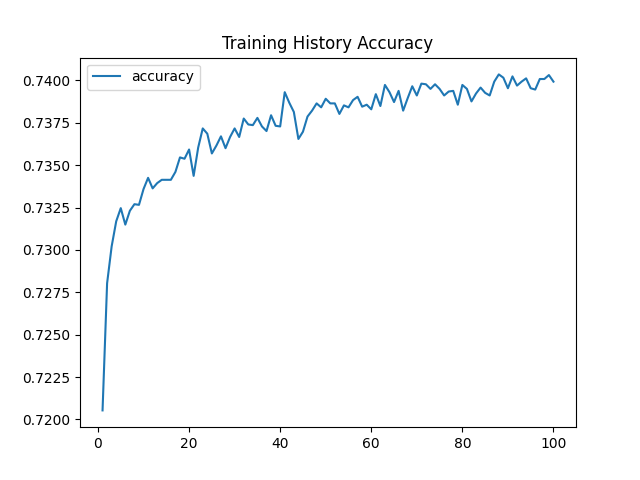
* There are 43 input dimensions.
* The first hidden layer contains 80 neurons
* The second hidden layer contains 30 neurons
  + The relu activation function is used for both hidden layers.
* The output layer contains 1 neuron
  + The sigmoid activation function is used for the output layer.

## Training

The model is trained using the scaled features training data, and target training data.100 epochs are used.

## Evaluation

*Figure 5 – Loss and Accuracy evaluation for testing data*



*Figure 4 – Loss and Accuracy plots for training data*

Figure 4 displays the performance of the model on the training data:

* Accuracy: ~74%
* Loss: ~53.5%

Figure 5 displays the performance of the model on testing data (which it has not seen before). As expected, the performance is worse when compared to the training data (which the model has seen before).

* Accuracy: 73.0%
* Loss: 56.1%

# Optimisation

In order to achieve a predictive accuracy higher than 75%, the model needs to be optimised.

The following measures were taken to do this:

* Reducing the number of features:
  + The number of bins in “APPLICATION\_TYPE” was changed, by adding “T10” to “Other”.
  + The number of bins in “INCOME\_AMT” was changed, by merging “5M-10M”, “10M-50M”, and “50M+” into 1 bin called “5M+”.
  + The “STATUS” column was checked for outliers, and thusly removed
  + The “SPECIAL\_CONSIDERATIONS” column was checked for outliers, and thusly removed
* Increasing learning:
  + The number of neurons in the first hidden layer was increased to 100.
* Reducing overfitting:
  + The number of epochs was lowered to 80.

By reducing the number of neurons and epochs, the risk of overfitting the model is decreased; the model does not become overly reliant on the training data and will be more able to fit new unseen data. By increasing the number of neurons in the first hidden layer to 100, to model is able to learn the training dataset more.

The process of striking a balance between overfitting and increasing learning required some trial and error in changing the neuron number and epochs.

* The activation functions were kept the same as relu is ideal for hidden layers, and sigmoid is ideal for the output layer.

Despite all these modifications, we were not able to improve the performance of the model to 75%.



*Figure 6 – Loss and Accuracy evaluation for testing data after optimisation*

As shown in figure 6, after the optimisation process, the accuracy was increased to 73.1%.

As the data is tabulated, it might be better to use a random forest classifier; python also enables us to easily identify which features are of importance when using a random forest classifier.