The purpose of this analysis is to assess historical data from Alphabet Soup to create a model that will identify future applications most likely to be successful in their ventures. The historical data received from Alphabet Soup included 34,000 organizations that had been funded in the past, data about the organizations and the classification of whether or not each organization was considered successful after funding.

**Results**

**Data Processing**

Model Target

The target for my model is determining will an organization be successful. This information is key in helping Alphabet Soup maximize ROI for their funding.

Model Features

The finalized model used 45 features, which were generated by embedding the following columns:

* Application Type
* Affiliation
* Classification
* Organization
* Income Amount\*
* Special Consideration
* Ask Amount\*

\*Ask Amount and Income Amount were combined into categories, and the $0 amount category was dropped.

After rigorous testing it was determined that the model worked better with all of the above information. I tried multiple variations of removing the above columns, or removing embeddings from the above columns, but the model was most accurate with all of the information.

Removed Model Features

The following information was removed from the model based on being company information with no value in being classified for a neural network:

* EIN
* Name

The following categories were dropped after testing revealed the model was more accurate without them:

* Status
* Use Case

Both of these categories had lopsided data: Use Case was classified 82% of the time as “Preservation” and Status had 34,294 instances of 1 and only 5 instances of 0. This helps explain why the model worked better without these categories.

**Compiling, Training and Evaluating the Model**

In order to avoid overfitting on a model with 45 features I attempted to keep the number of neurons, low and focus on using more hidden layers. The total number of neurons used was 320, spread out over eight hidden layers which used the following activations:

* Relu
* Leaky Relu
* ELU

Accuracy

Despite countless variations or both neural network setting and variable refinement I was not able to reach .75, plateauing consistently at about .72.

Steps Taken

My steps taken to improve accuracy fall into the following categories:

* Refining variables by creating bins.
* Embedding variables, testing them in a neural net and removing unhelpful columns
* Testing individual embeddings such as “other” categories, “$0” bins
* Changing the number of layers in the neural networks
* Changing the neurons in each layer
* Changing the activations
* Changing the loss function
* Changing the optimizer
* Changing the EPOCH count

Each step focused on changing one of the categories at a time and grading the result.

**Increased Accuracy after Refining the Model**

The initial run of the model produced a loss of 0.6024953722953796 and an accuracy of 0.6983965039253235.

A screen shot of a computer code

Description automatically generated

After optimization the model produced a loss of 0.6223226189613342and an accuracy of 0.7198250889778137.

A screen shot of a computer code

Description automatically generated

**Summary**

By using relatively simple classification information from Alphabet Soup the neural network model was able to achieve a high level of accuracy in classifying success.

Variables such as Affiliation and Organization are useful, but missing information includes the quality of the idea, existing product, product team, founders, mentors and so on.   
  
Because that data is missing and because the model has un upper limit below .75 I believe that this Neural Network can be a very useful tool in narrowing down the types of organizations that have been successful so that Alphabet Soup can narrow down what companies it wants to spend time researching and meeting with.