Daylan Quinn Report CS-461 Project 3

This report is separated in 23parts – Data Cleaning, Model, Results

To begin I used the Pandas library to take in the csv data as a Pandas Data frame. The data frame enabled me to better transform some of the column data for better training. I used the LabelEncoder from sklearn to encode words in the data as numbers for easier learning by the future model. From the sklearn library I also used OneHotEncoder to one-hot encode my data and MinMaxScaler to normalize my data. Cleaning the csv data was the hardest part for me as I wanted to ensure that everything was put together well so my future model would be as efficient as possible.

Once my data was properly prepared, I began to split the data to be trained. I one-hot encoded the response data and separated it from the data to be trained since it is the labeling that helps the training data know what it is training for. In the beginning I thought I had 100% accuracy but then realized I did not take the response data out of the training data and it was being trained with the answers. The data was then split for 85% training and 15% testing and validation. I used Keras from Tensorflow as Keras is suited for Deep Learning materials and I used a Sequential model to build a linear stock of layers. My first layer has 25 neurons, and my hidden layer has 50 neurons, both using ‘relu’ activation. The layers and use of ‘relu’ activation were determined based on what worked the most efficiently through trial and error. For my output layer I used the ‘sigmoid’ activation function as it is used more for binary outputs. I then trained my data on the model with 5 epochs, batch size of 128, and passed in my validation data.

With my model trained I began inspecting the results. The accuracy would be around 87-88% through different trials and the loss was around 26%. I used matplotlib to graph the accuracy and loss and they both seemed to have nice curvature to let me know the model was improving and on the right track. Since the data learned from so many results of response 0, I noticed it seemed to favor a prediction with 0 instead of 1 sometimes. The model seemed to predict almost every 0 response correctly and the occasional 1 response correctly. In the end it seems to be a decent classifier that can mostly predict when a 1 response would be possible. Attached below is the graphs of accuracy and loss.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated