Andrew Pascual ID: 219494575
Joshua Nguyen ID: 219517364
Bryan Ho ID: 219405057
CSC 180
Chen
26 February 2021

Project #1 Report

Problem Statement

In this project, we were tasked with gathering three to five arbitrary businesses from a full Yelp dataset. Additionally, these three to five businesses must have at least twenty reviews and will be tested with different neural network implementations in TensorFlow. After gathering data as to which neural network model was the best fit for predicting the review stars on yelp, the next issue to tackle was representing such businesses alongside their respected names, predicted ratings, and actual ratings. The last obstacle given was to report the RMSE of the best neural network model obtained and plot the corresponding lift chart.

Methodology

We began this project by downloading and gathering the dataset from Yelp. This download contained five JSON datasets including the reviews, businesses, check ins, tips, and users. However, for this project we only needed to work with the reviews and business files. In order to work with these files we then converted each JSON file into two separate tabular formats that Pandas could read. With the given sample code from the assignment instructions, this conversion was possible and we then established the two starting dataframes which held the reviews and business information. With the reviews and business information we then merged the files into one dataframe by aggregating the reviews to each business. Moreover, we cleaned some of the data by removing any businesses that had less than 20 reviews. Once this was completed, we did one final merge of the aggregate reviews and business dataframes in order to get the most important dataframe for the rest of the project 'df_review_business'.

The next step in our project was getting the TF-IDF and vectorization completed. This will allow us to extract features and deliver the uniqueness from the business data. As learned from the lectures, the TF-IDF is a great tool for retrieving unique words from a document and sorting the unique words in a neat order. From this step we were able to retrieve information revolving around the compressed matrix of TF-IDF data and use it as our X and use the stars from the review business dataframe as our Y. Following this, we used such variables towards the splitting of data and then we were able to train and test 20% of the full data. Then, with our split train and test data, we used such variables into the numerous neural network implementations in TensorFlow. By tuning the hyperparameters around different activations, number of layers, neuron counts for each layer, and optimizations, we were able to distinguish each model and find the best model for predicting the yelp review business stars.

Finally, after comparing each model we came to the conclusion that the best model was the combination of 'sigmoid' and 'adam' which was our model 3. We chose this as the best model because of the lowest RMSE score. Now that we have obtained the best model, we wanted to use this to obtain predicted ratings of our five arbitrary businesses and hold it alongside the actual ratings, names of the businesses, and the business ID. After showing how accurate the prediction was, the last challenge was to plot the lift chart for our model and data. This portion was a large struggle that will be discussed in the Task Division and Project Reflection section. Ultimately, we resolved the issue and managed to complete the last task required for the project.

Experimental Results and Analysis

After executing our methods, our final results are:

Results of changing the activators and optimizers :

During the hypertuning process, we mainly focused on the effects of changing the activators and optimizers for our data. For each process there were 4 layers, with 100, 10, 5, and 1 neurons per layer, in our model. We also set the patience to 5 and the epochs to 100. After running the different tunings, we found that model 3 with sigmoid and adam to be the best models for our data. To note, by changing the patience around to the values of 2, 3, and 5 we found each RMSE to change and have a different best model for each patience and we ultimately stuck with 5 patience.

Results of changing the layer count to have less than 4

We found that the accuracy was increased because the RMSE was lower by .0002 however, the speed was slower because there were more epochs. The epoch count was 18 before EarlyStopping and had a .2704 RMSE.

Results of changing the layer count to have more than 4

We again tested on model 3 to see the effects of layer count. We added another layer with a dense of 2 and that increased the RMSE slightly. The number of epochs ran before stopping was 12 but ultimately, the additional layer helped run the data faster and only had a slightly higher error. Therefore, by increasing layer count to the hyperparameter then the process is slightly faster but will have a lower accuracy and by decreasing layer count then the process is slightly longer but has a better accuracy.

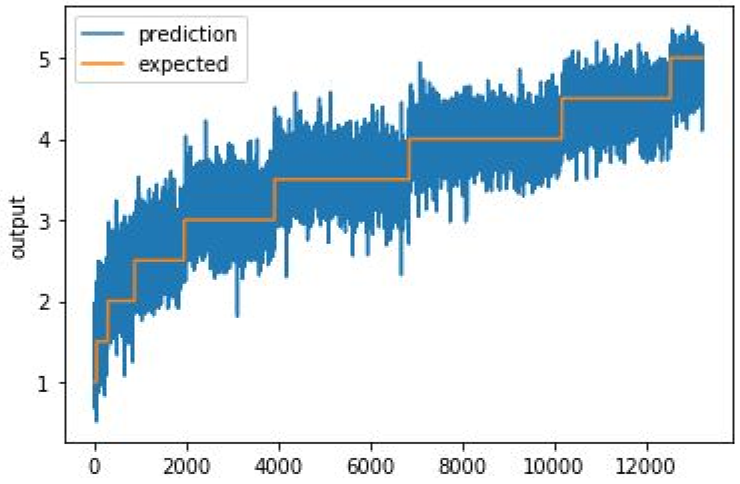
Results of increasing each layer’s neuron count by 10 except for the last layer

We tested on model 3 to see the effects of neuron count and found that by increasing the neuron count, we achieved a lower accuracy because the RMSE was higher at .2739 but a slightly faster process because it had 9 epochs before EarlyStopping.

Results of decreasing each layer’s neuron count by half except for the last layer
We tested on model 3 to see the effects of less neuron count and found that by decreasing the count, we achieved a higher accuracy because of the lower RMSE of .2677 but a longer process

because of the 23 epochs before EarlyStopping. Thus by an increase in neuron count then we give up some accuracy for better run times but we can gain accuracy for slower run times by decreasing the neuron count.

Lift chart and RMSE of the best model:



```
Score (RMSE): 0.27064694962174707
```

Task Division and Project Reflection

We divided these tasks equally in order to complete this project efficiently. Andrew was tasked with performing the training and testing split, removing all the businesses with less than 20 reviews, using TF-IDF to perform feature extraction from review texts, implementing EarlyStopping when using Tensorflow, reporting the RMSE, plotting the lift chart. Bryan was in charge of changing the activation, layer, and optimizer hyperparameters to record how they affect the performance, and ran the tests and combinations to find the optimal pathway. Joshua was in charge of writing the report and converting the JSON files to CSV.

There were many challenges we faced while completing this project. One obstacle we faced was extracting the dataset properly, which led to another challenge we encountered, which was converting the dataset JSON file to CSV. These two initial challenges were caused by technical and hardware difficulties. Another challenge we encountered was fine tuning the hyperparameters, because the function of EarlyStopping was not initially met so we did trial and error over a long period of time in order for our systems to achieve the function. Additionally, when all models were compiled on separate hardware, we found that the RMSE scores and the runtime varied. We therefore concluded that hardware implementations make a difference in both the compilation time and the actual performance of the model, and opted to use a single set of hardware to perform the rest of the computations in order to provide standardized data. In addition, another struggle we stumbled upon was the process of plotting the lift chart for the best regression model we acquired. We had a difficult time trying to output the chart because of a flatten function, but we were able to fix the error by removing it in the initialization of the chart_regression code block [114]. Ultimately, these challenges only brought out more knowledge to the project and assisted our learning of neural networks.

Although we gave ourselves a week to start on the project, we learned that starting the day the assignment is given is a much better strategy. Another lesson would be that the root cause of bugs may be simpler than we expect; for example, a bug in which Python told us that the 'Series' attribute had no attribute 'Flatten' which caused us trouble, as we attempted to graph the results of our model. However, the resolution to this bug was simply to remove the "Flatten" attribute at the instance we define 'chart_regression', which came as much relief after we tested much more convoluted solutions like completely converting the variable to a different type. Another lesson learned would be the importance of TF-IDF and effectiveness it holds alongside neural network models. Conceptually, we learned how to train models and test them so that they are able to predict values based on their training and testing.