Mini Project 1

Ezekiel Sung: 219507107
Hung Truong: 301035748
CSC 180-01
Due:2/26/21

Problem Statement

Our goal for this project is to predict a yelp business's star rating based off of all their reviews using neural network implementation in TensorFlow.

Methodology

The tools and libraries we will use consist of TensorFlow, sklearn, Numpy and Pandas. Using these resources, we will first data pre-process and clean up the data by dropping businesses with less than 20 reviews and unwanted columns. We also dropped any rows with null values to make our dataset more accurate. Then we have to deal with reviews.json because it is a very large file, we will only grab the first 500,000 reviews and drop the columns that are unwanted. We also will clean the reviews by removing stop words. Next, we will merge these two datasets by business_id and group all the reviews together by the same business. Then finally we will use TF-IDF vectorization for feature extraction to extract the top 1000 words with a minimum frequency of 10 in the reviews and convert it to an array so that we can train our model.

Before we can train our model, we first will split our data into 20% test and 80% train. Since this problem requires a regression model, we will use Neural Networks in TensorFlow and adjust the different hyperparameters to see the results. These are different hyperparameters:

- Activation: relu, sigmoid,tanh
- Number of Dense Layers and Neurons in each Layer
- Optimizer: adam, sgd

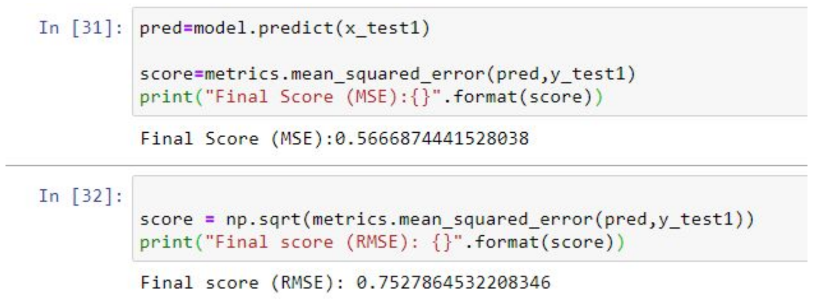
Along with these hyperparameters, we will also use earlystopping to prevent overfitting the model and use checkpointer to save the best run and to avoid local minimum.

After having our model, we will adjust the different hyperparameters and record our different findings to see what will result in the lowest Mean Squared Error and Root Mean Squared Error. After finding the best hyperparameters, we will graph the "ground truth" vs "prediction" and also show the lift chart for our best model.

Experimental Results and Analysis

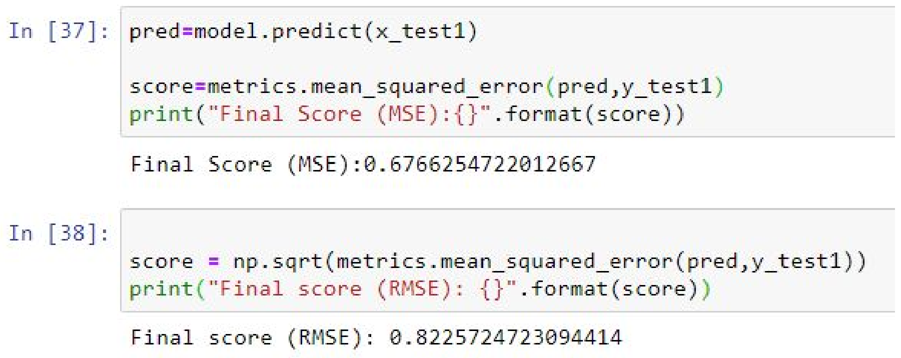
Activation: Relu

After trying out different hidden layers, neurons for each hidden layer and optimizer for Relu, we have found that relu produced the highest mean squared error and root mean squared error. With optimizer adam and regardless of how many layers and neurons, the results are very similar.



```
In [31]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.5666874441528038

In [32]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.7527864532208346
```

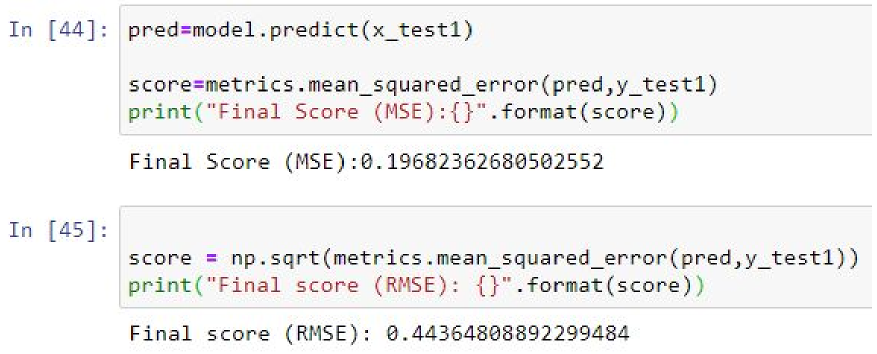
Tuning the different hyperparameters all gave similar results of around .55 MSE and .75 RMSE with the adam optimizer. With sgd optimizer, the results were about .1 higher than adam as shown below.



```
In [37]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.6766254722012667

In [38]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.8225724723094414
```

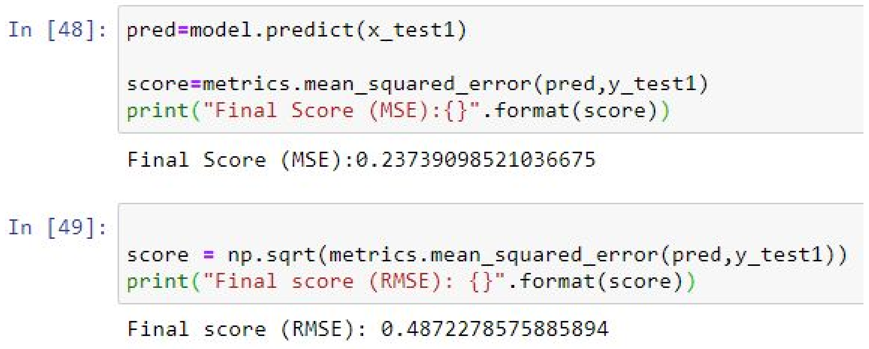
Activation: tanh

We found that tanh gave the second lowest final score. We first tried tanh with adam optimizer which gave similar results to sgd optimizer. Right from the start with only 1 hidden layer and 10 neurons, we already have much better results than relu.



```
In [44]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.19682362680502552

In [45]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.44364808892299484
```

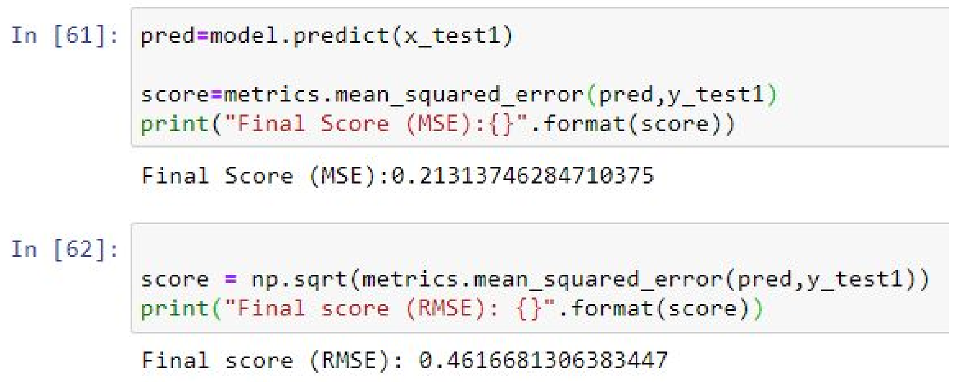
Keeping the model with 1 hidden layer but increasing the neurons to 50 instead of 10 did not result in a better final score.



```
In [48]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.23739098521036675

In [49]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.4872278575885894
```

Increasing the amount of layers also did not result in a better score regardless of the amount of neurons.

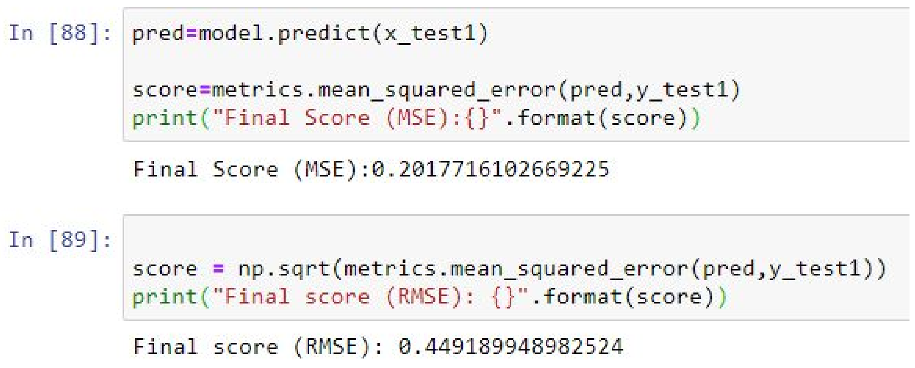
Then with sgd optimizer, regardless of the amount of hidden layers and neurons, the results were better than relu, but not as good as tanh with adam optimizer. The average MSE for sgd optimizer was .22 and RMSE was .46.



```
In [61]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.21313746284710375

In [62]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.4616681306383447
```

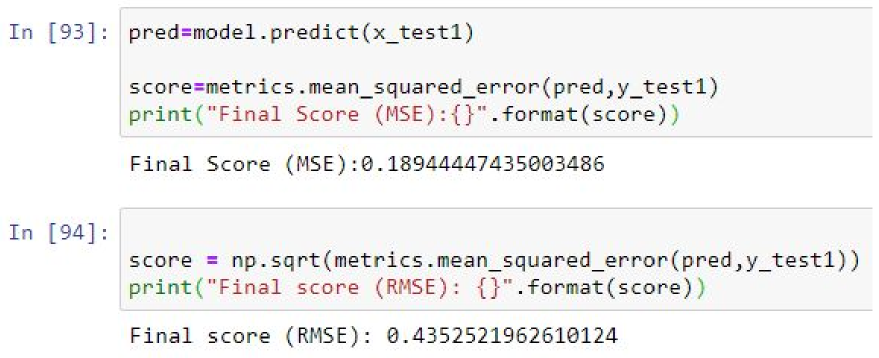
Activation: sigmoid

We found that with sigmoid and adam, it works better with just 1 hidden layer and more neurons. The more neurons the better, but it caps out at about 50 neurons. Below is sigmoid and adam with many hidden layers:



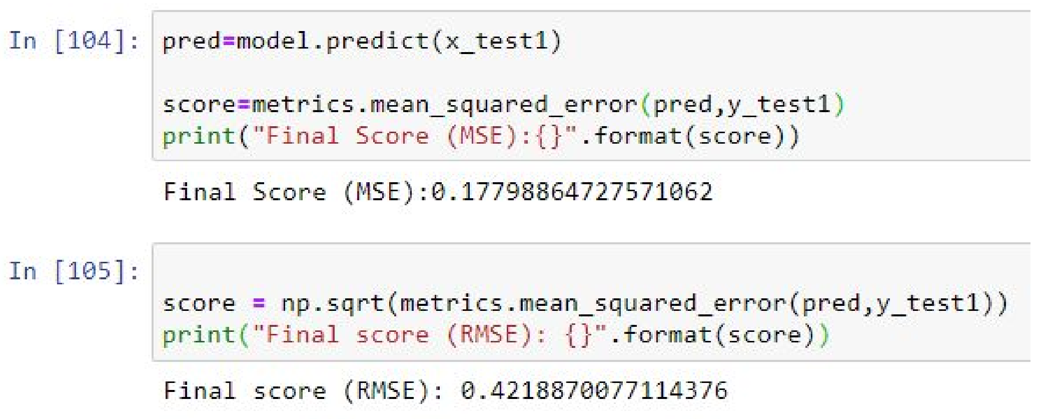
```
In [88]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.2017716102669225

In [89]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.449189948982524
```

Then with just 1 hidden layer and 50 neurons:



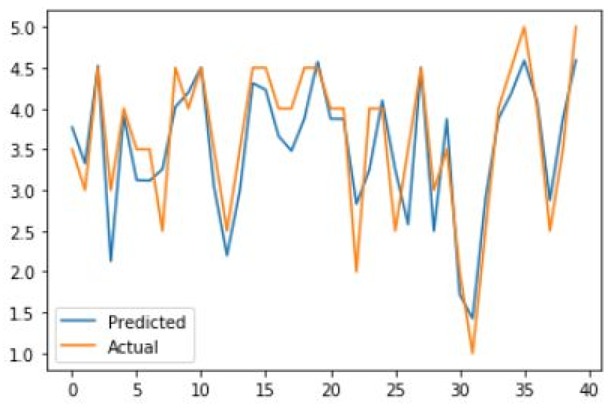
```
In [93]: pred=model.predict(x_test1)

         score=metrics.mean_squared_error(pred,y_test1)
         print("Final Score (MSE):{}".format(score))

         Final Score (MSE):0.18944447435003486

In [94]:
         score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
         print("Final score (RMSE): {}".format(score))

         Final score (RMSE): 0.4352521962610124
```

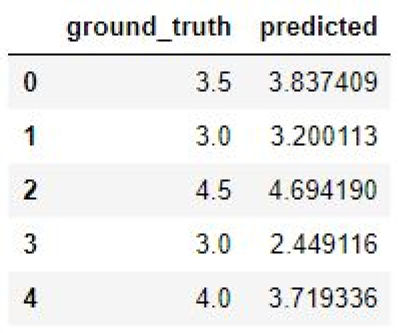
Our best results were given by sigmoid with sgd optimizer. Although the results of both sgd and adam are very similar, we were able to have best results with sgd. The best hyperparameter settings we found were sigmoid, sgd, 1 hidden layer and low amount of neurons. Here are the final scores of this:



```
In [104]: pred=model.predict(x_test1)

          score=metrics.mean_squared_error(pred,y_test1)
          print("Final Score (MSE):{}".format(score))

          Final Score (MSE):0.17798864727571062

In [105]:
          score = np.sqrt(metrics.mean_squared_error(pred,y_test1))
          print("Final score (RMSE): {}".format(score))

          Final score (RMSE): 0.4218870077114376
```

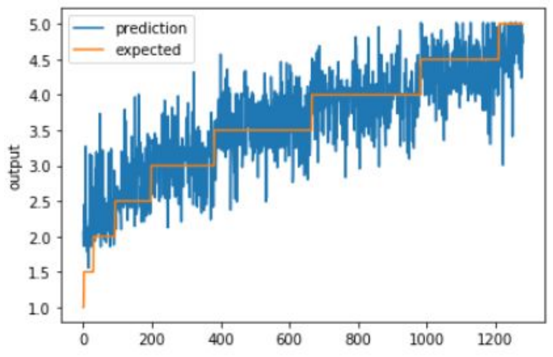
With the best model here is a graph and table of ground truth versus predicted.



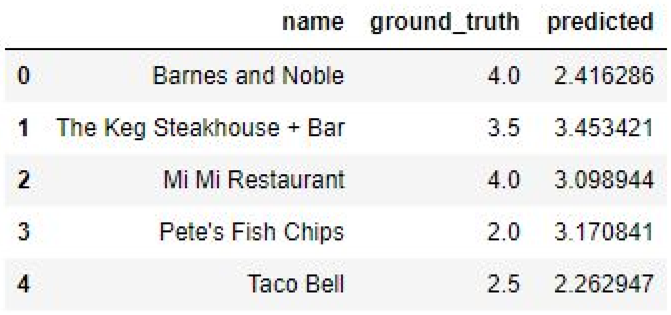


	ground_truth	predicted
0	3.5	3.837409
1	3.0	3.200113
2	4.5	4.694190
3	3.0	2.449116
4	4.0	3.719336

Here is the lift chart of our model with the colors of predicted and actual reversed because it was hard to see the expected line with the amount of data we used. So in this sorted lift chart, the prediction line is blue and the expected line is orange. The lift chart shows that there is a gradual increase in prediction that follows the expected line.



We tested our model with 5 different businesses and we tried to get different categories to test our model. We created our own data set filled with all the names, reviews and star ratings. After loading up the test data and using our model to predict the ratings, the below chart is our results. To our surprise that the predicted value is not as close to the actual value. It seems like our prediction is off the further away from a star rating of 3. We think this may have happened because maybe yelp's average rating is around 3 so our training data set has more reviews for 3 stars versus 5 stars and 1 stars. Next time we can maybe try to filter out our data set and have an equal amount of businesses of each star rating to improve our model even more.



	name	ground_truth	predicted
0	Barnes and Noble	4.0	2.416286
1	The Keg Steakhouse + Bar	3.5	3.453421
2	Mi Mi Restaurant	4.0	3.098944
3	Pete's Fish Chips	2.0	3.170841
4	Taco Bell	2.5	2.262947

Task division

Hung was in charge of research, figuring out what we needed to do and how to do. He also created a task board on jira for this project. Then he was in charge of showing the findings from our model through graphs and tables. After having all of our data, he also made the report and recorded the video.

Ezekiel was in charge of setting up the data. He loaded the data, cleaned the data, merged the data and used TF-IDF vectorization. After prepping the data, he was in charge of splitting the data and training the model with different hyperparameters. Using earlystopper and checkpointer, he tested out different hyperparameters settings and took note of the results of each trial.