Justin Singh
ID: 301690181
CSC 180 - Intelligent Systems
Project 1
Due: Sep 30 at 11:59PM

Problem Statement:

As a developer, I aimed to predict business star ratings based on the review text for each business. While solving this problem, I ignored all businesses that had less than 20 reviews.

Methodology:

To solve this problem, I used regression. I chose regression because it provides numerical output values which would help for when I did my fitting. I had to use tfidfVectorizer to convert the review text into vectorized values so I could feed the values into my data model. I used neurons for my data model because they are good with regression problems. I chose relu as my activation for my layers and sgd for my optimizer.

Experimental Results and Analysis:

Activation

relu	**sigmoid**	**tanh**
Research showed that relu is the best activation to use for regression problems. After running through multiple trials, I would agree.	Right behind relu. I found that RMSE values changed by 0.1 when toggling between relu and sigmoid	I found this one to provide the worst RMSE values. I feel this activation is not meant for regression problems.

Optimizer

adam	**sgd**
Originally I found adam to be more successful with lower RMSE scores. However, once I started using optimizers sgd calculated RMSE scores about 0.2 lower then adam. optimizers for adam constructor didn't seem to help any.	Performed the best. I receive RMSE scores about 0.2 lower then adam. To achieve this I had to use the optimizer constructor and adjust the learning rate.

Number of layers and neuron count for each layer

I used a total of 2 layers. One input layer and one output layer. I found that additional layers made the fitting process longer and it did not help RMSE scores. For the input layer, I had a total of 5 neurons. Any lower than 5 resulted in poor RMSE values. The higher I went the more the RMSE scores didn't change. Once I tested past 50 neurons and the RMSE scores were extremely poor. I found the sweet spot to be around 5 neurons as a result. Since we are dealing with a regression problem, I chose only 1 neuron for the output

Task Division and Project Reflection:

I worked on this project solo. Therefore I am responsible for all of it. One of the most challenging portions of this project was understanding how to use the vectors of text and the reviews and split them. After reviewing notes, I realized that x input is always the data you are comparing and the y input is what you want to predict. It became clear that x was for the review text and then y was for the stars. An important note here is that both the reviews and the stars had to be numpy arrays when fed into train_test_split. That was a headache trying to figure that out. Another painful portion of this project was optimizing to try and get a lower RMSE score. Research showed that there is no set standard for calculating how many layers or neurons are needed. There were only best practices. One of the best practices I researched was that regression problems perform well when there is only one input layer and one output layer, no hidden layers. This worked out well for me.