Homework #1 Report on Amazon reviews
Name: Hamza Khanane

Team Name Registered on miner web-site: Pak97
Rank & Accuracy: 77% Leadership board No. 34(10:44 PM 10/18/2018)

This homework assignment took a lot of time mainly because of the research I had to do to implement my algorithm. At first, I didn't have an idea of how or what KNN algorithm worked like. After a decent amount of research and watching YouTube videos, I finally understood the main idea behind this algorithm. The first step which I took to was to read my training data and test data files line by line and stored each review into their respective lists. The next step to clean the data first so that all the useless words and punctuations are removed. I removed the stop words using a NLTK library and ran a loop through my reviews. While removing the stop words I also removed the punctuation marks using a string function. The other measures I took to clean the data was to perform stemming using a Port Stemmer which is also a part of the NLTK library. This approach was taken to clean the data from both the test file and training file.
At first I created a bag of words using the training set but after doing some more research, that approach did not make sense to me and I expected a bad runtime so I decided to use another approach. I started doing some research on the TFIDF vectors and the cosine similarity. The approach started to make more sense to me and I decided to go with it. I imported the TFIDF library and made a tfidf object. After that, I transformed my training data and test data into two different matrices. I ran a loop to compare one element from the test matrix to the entire training data reviews matrix which totals up to 18506 elements. I used each row from the matrices to calculate my distance using the cosine similarity function which I also used from one of the imported libraries. As soon as I got my calculated distances I started appending them into a new list. My next step was to decide on how I would be implementing my KNN algorithm. I decided on getting my k largest distances from the similarities list in which I had all my 18,506 distances. I sorted my list using the sorted() function so that I can easily get my largest values. The list was now sorted in a descending order and I just ran a loop to get my k number of largest values. I started appending those largest values into a different list. Before cleaning my data, I had also created a separate list just to append the +1's and -1's so that I could refer it to later and check for my positive negative review indicies. As soon as my list had the k number of largest values I started checking for these indices of these largest values in nonsorted similarities list. Getting those indices helped me get to the same index and in my plus and

minus list which contained +1 and -1's. I appended those scores into a different list which I later used to get the majority vote. The majority vote was basically one of the parts of my KNN algorithm. I started with k=13 and my accuracy rate was 55. I started going higher and implemented it using k=25 which improved my accuracy to about 76%. I tried to give it the last shot since I still had one more submission left, I changed my K to 19 and my accuracy rate jumped up to 77%. Though it wasn't just the K which improved my accuracy I also made some changes to my tfidf vector.

Overall I used this approach because it made the most sense to me. The bag of words approach also had made sense to me before but then I had to scratch it because I felt that the implementation of that would become really complicated and I would have a bad runtime. The tfidf and cosine similarity seemed like the easiest approach for me but it did not work as accurate as expected.

The cosine similarity basically checks how close the two texts are. The cosine similarity takes the vectors related to these texts a throws out a number telling us how similar they are. The cosine similarity does not care about the meaning of these texts. Even if they don't mean the same thing or talk about the same thing, cosine similarity will still work and return a number which gives us an idea about similar these texts are. I still think that it might be not so accurate when using it to find how similar the sentences of these reviews are. I am sure that every review is different in its own way and every person uses different types of languages/words to describe their emotions and reviews about the product.

This accuracy metric is one of the most suitable ways to evaluate this assignment as the data which has been provided to us has equal numbers of reviews belonging to both the classes. For example, there is a equal number of +1's and -1's reviews in our training data . This method would be really unsuccessful if the data given to us did not have equal number of samples given. For example, if we had 95 percent of positive reviews and 5 percent of negative reviews(95 +1's and 5 -1's) then anyone could've gotten a 95 percent accuracy rate just by predicting that all of them are positive. So it is suggested to use some other kind of method to evaluate the classifier in these type of cases.

I am sure that I would have done better on this homework if I started a bit more earlier and did some more research to get a better idea of what implementation would be the most accurate. Anyways this was the first HW so I feel that I learnt a lot of new things which I never knew existed and I am excited to learn more in the next homeworks and obviously learn from my mistakes and research for a more accurate implementation.