Emma Peatfield 009192534

Rank: 19 F1: 0.7666

**Program 1 – Medical Text Classification**

At the beginning of this assignment, I had 2 files; training data file and a test data file. Both of these files contained medical entries, but only one of the files had the entries placed with a category 1 through 5. The goal of this assignment was to classify these test medical entries and place them with the correct category.

To start, I had to preprocess my data. The first step was to separate the already classified entries and their class, so that I could focus on the actual text of the medical entries. Once this was done, I wanted to process my data a little so that I could focus on comparing the words within the entries that were the most important. First, I removed any numerical values or characters within the set. Then, I removed functional words, such as “the, a, as, and, etc.”, by using the stop words package from nltk, as well as any punctuation, symbols, and characters that were unnecessary. Finally, I stemmed each word, so that I could see the basic root of the words, such as “sleep” and not multiple versions of the same word, such as “sleeping, sleep, sleeps.”

Once I had preprocessed my training and testing data I started writing my algorithm for k-Nearest Neighbor. I decided to use TF-IDF and cosine similarity measures between the medical entries to enable my code to predict the most likely category for each class. TF-IDF, or Term Frequency-Inverse Document Frequency, would enable me to measure how important a word is to a specific document or collection of documents. Thus, for each document, I could find the most important words and how important they also are to the set of documents. To do this, I calculated the tf-idf for both the training data and the test data and stored both of these in matrices labeled *wmatrix* and *testmatrix*.

After calculating the tf-idf, I compared these two matrices, document-by-document using cosine similarity. Thus, I took the dot product of the two vectors (or rows) and then divided that by the magnitude of each of the two vectors. This value was then added to a matrix that would contain the similarities between each training document and test document.

Once the similarities were stored, I was able to find the k-nearest neighbors for each of the test documents and from this, I was able to output the most likely category of that document into a text file of predictions. I did this by creating a vector that stored the most likely class for each document and finding which class was the most common for each document, i.e. taking the mode of the vector.

I chose this methodology because we had covered TF-IDF in my Large Scale Analytics class and we had discussed cosine similarity in both that class and this Data Mining class. After having learned about them, I wanted to learn how to implement these two on my own to process my data. It took quite a while to get a solid code working, but I managed. However, the downside is that my dot product takes a very long time and I've been trying so hard to reduce the time, but I have only been able to reduce time in creating the matrices, but not in the calculations section.

With that, a fair warning with my code is that it runs for a very long time. The preprocessing and early steps only take maybe a few minutes, but it takes about an hour for the matrices to undergo a dot product and cosine similarity calculation. The code works and outputs the prediction values that have been uploaded to CLP, but it takes a very long time. I hope that that is something that I can improve with my code throughout the semester because I really want for this code to run faster and have a higher accuracy. Besides the time, I don't think there are any other specific instructions for running the code, it should work just fine, it just takes a while.