1. Team Name Registered on miner web-site.
2. Rank & Accuracy score for your submission (at the time of writing the report).
3. Your Approach
4. Your methodology of choosing the approach and associated parameters.
5. Describe how the metric Accuracy is computed. Which application will this be an unsuitable metric?

**Team Name/Personal Name on Miner:** tnguy145

**Rank & Accuracy score:** 57%, 75th place

**Approach**:

There are multiple approaches I have chosen for this project. I first did data cleaning and data separation using Python Pandas and numpy packages. Positive and negative comments are being separated into 2 different files separated by a new line. I then go through all of the words in each of the file and count it repetitiveness. After that, I used Java to build the classifier. In Java, I make my first attempt by storing the positive and negative words into a TreeMap (which is sorted) for better KNN performance. I then use those bag of words to go through all training data and gives it a scores and store it in a HashMap. Test data is also being read and given score the same as the training data. All of the comments by now will have a positive, negative, +/- status stored in its values array. The key of the HashMap is the average of the positive and negative counts. After that, with the calculated positive and negative points of the train data set, I will get it average score, and apply KNN to get values that has the nearest average number. This method yields the highest of 49% accuracy.

The second approach I have chosen for the project was to add Euclidean distance and Cosine Similarity. I used positive score and negative score of each of the HashMap key (from train data set) and the current comment from test set data. I then iterate through the whole train data set and give it a Euclidean and Cosine similarity distance with regards to the current comment. After that I store those data into a TreeMap where it automatically sorts the key. Since it’s sorted, the result can be retrieved based on K and pick by using majority voting. The method sees the accuracy increase to 52% on Cosine Similarity and around 51% on Euclidean Distance.

The third approach was to add TF-IDF to the score calculation metric to assign negative and positive words with a more accurate score. There was a slight increase to 54% using Euclidean distance combine with TF-IDF while K=7. Cosine Similarity combines with TF-IDF yields a 57% accuracy while having K=5.

After multiple tries, it seemed like K=7 is the optimum K number for my algorithm as it usually yield a highest accuracy score.

**Choosing approach and parameters:**

Since this is the first data mining project, I want to migrate myself by starting with easy approach. As described on **Approach,** I took 3 different approaches in this project to see what would works best. I also try multiple different K variable, starting with 1 and 3 and 5 and goes all the way up to 49 to see what offer better result. After multiple tries, it’s clear that keep K at around 5 and 7 is the optimum choice for the project.

**Accuracy Metric:**

I calculate my accuracy based on the training set. I parse the data, only takes the +1/-1 format and compare it with my output to see the accuracy. I have also compared my result with the format file. This accuracy metric is not suitable for application that has test set where it’s a little bit different from the training set. If using this accuracy alone would also make the algorithm over fitting.