# Implementation of K Nearest Neighbors learning Model

Miner User name : dumbbot
Rank : 89 rank
Accuracy Score : 76 score
Gmu ID : mkhan96@gmu.in

## My approach

I have implemented the KNN using Tf-Idf and cosine similarity.
After preprocessing and lemmatizing the training and test data, I create a term frequency vector for training and test data. Once the respective vectors are created I find the similarity between each of test reviews and the training reviews as a sparse matrix given by cosine_similarity function.

> The similarity value of test case 'x' and a training review 'y', is present in the cell corresponding to x,y of the sparse matrix.
> Thus, each row of the sparse matrix represents a test input's similarity with all the training reviews.

I then find the 'k' highest similarities from each row and corresponding classes (+ve or -ve feedback) associated with those reviews which have resulted in the high similarities. Then classify the test input on the basis of majority.

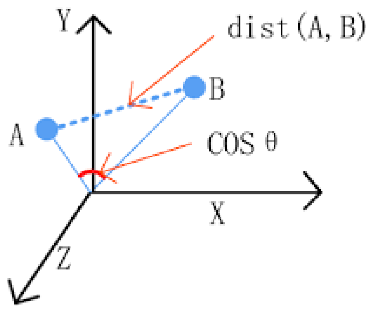
## Why the approach was chosen

Lemmatization over Stemming : Lemmatization considers the meaning and context behind the word before converting it to its base form. It is a costlier approach in terms of time efficiency but since we are dealing with similarities between strings, it is a better approach.

> Lemmatization handles matching "car" to "cars" along with matching "car" to "automobile". Stemming handles matching "car" to "cars" .

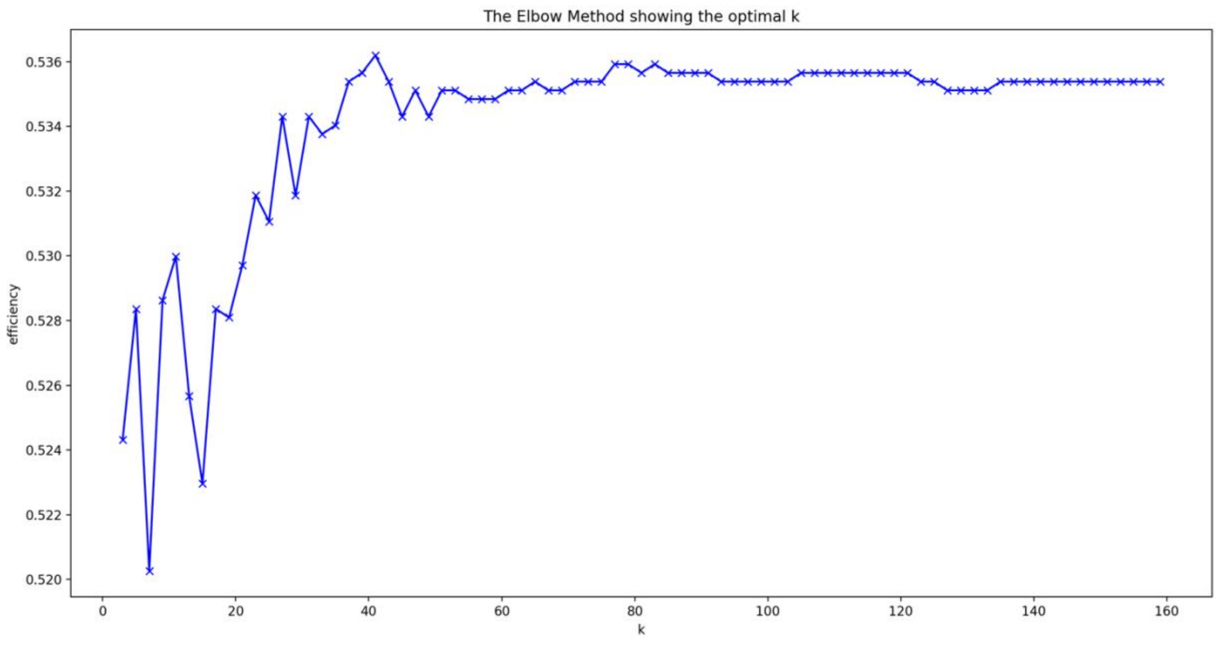
TF-IDF : After preprocessing and lemmatization I have a document with reviews containing just the base form of the words used. The test and training data is normalized to some extent. Now I need to convert this document into numbers or scalars I can use my distance/similarity function upon. Tf-Idf gives every word a number and also based on its importance to the document normalizes it further.

Cosine Similarity: Cosine similarity tells how similar is the word based on how angularly close they are in space. Cosine similarity considers the orientation of the two inputs for their similarity rather than simply calculating the distance.



## Optimal value of k

To find the optimum value of k(no. of neighbors to consider for classification) I shuffled and split the training set into a ratio of 80:20 and used the 80% one as a training set for my 20% one. Then compared the predicted results with actual classification of those 20% reviews. Following are the results:



The highest accuracy achieved on 80/20 split was at k = 41, of 53.62%
For a 70/30 split the max efficiency was reached at k = 19, of 53.47%
**Miner** score on both cases was **75**

The above accuracy was on majority selection where if the number of +1 class reviews are higher amongst the k neighbors then the test input is given +1.

For weighted selection i.e. I multiplied the similarity value with the class value (+1/-1) and then summed it up for the k neighbors. If the sum is positive, then +1 class is assigned to the test input.
I ran the program for k = 19 and 41 and got **76** score on **Miner**

So, the program will be running on weighted classification and on k = 41.