Name: Shrey Patel
ID: 012430652

Medical Abstract Classification using k-NN with min-epsilon

Submitted By: Shrey Patel (012430652)

Problem Statement: Develop a predictive model (kNN) that can be used to predict the label of given text document (medical abstract.)

F1 score (as seen in CLP system) : 0.6694
Rank: 33

Approach taken to solve the classification problem:

1. **Pre-processing**
   - Remove stop words and words with length less then 4. (Used nltk.stop_words). This will remove frequently used English common words that can interfere with document similarity measurement.
   - Convert document vectors to compressed sparse row (csr) metrices. This allows faster processing
   - Scale the csr matrix with idf (inverse matrix.)
   - Normalize the matrices using L2 norm.

2. **KNN Algorithm**
   - Used KNN algorithm with cosine similarity as a proximity measurement. After several test iterations, I chose **K value = 10** and **epsilon = 0.6**.
   - Build a custom k_nearest object that stores the similarity matrix for different computations between rows of test and train data matrix.
   - Used majority voting scheme, by weighting votes with distance metric (cosine similarity).

Methodology used for this approach:

- Major challenge is the imbalanced data set.
- Given training data has maximum classes of label ‘5’ and minimum classes with label ‘2’.
- To tackle this, I used under-sampling. In this scheme, we shuffle the data set, and randomly remove the extraneous classes in order to make this dataset more balanced.
- Experimentation: One of the most important part is to experiment with different values of k and epsilon on the re-sampled data.
- With some iterations, I found the k = 10 covers the relevant nearest (similar) neighbors most of the time.

Name: Shrey Patel
ID: 012430652

- Also, the neighbors with similarity < 0.6 were mostly of other class. So they are treated as outliers. Thus, we set epsilon = 0.6.
- K > 10 covers more noise than that is expectable.
- Epsilon < 0.6 might result in expected labels getting removed. (overfitting).

Instructions for running the application:

- Put the train.dat and test.dat file in the folder containing the python script.
- The script produces out.txt containing the output predictions.