**I. Score**

At the time of writing this report, I received an accuracy of 63% and ranked 30th within the public leaderboard. My registered team name on the miner website is **asdf\_dsa**.

**II. Methodology**

**a. Research and Data Pipeline**

During the first lecture, I understood the concept and procedures of k-Nearest-Neighbors (kNN); however, I had many unanswered questions such as, “How do I represent a word within a meaningful graph?”, “What do I use to process all of the words?”, “Should I keep all of the words within the reviews?”, “What is the Bag-of-Words model?” Despite these questions, thorough research within these four topics encouraged the creation of a pipeline. In order to understand and organize the complexity of the project, a data mining pipeline was created. The pipeline involved 5 stages: pre-processing, feature extraction, feature representation, cross-validation, and kNN classification.

**b. Libraries**

Learning the preceding stages to kNN classification was essential. With no experience in data mining, the goal was to avoid re-inventing the wheel for the project. Many libraries and models were available for each stage of the pipeline. For the most part, the available libraries would help in pre-processing raw text data, extracting features given a list of words, and representing features in an n-dimensional vector space. A few Google searches resulted in several libraries such as NumPy, Scikit-learn, NLTK, Gensim, GloVe, and FastText that fulfilled all of the requirements. After discovering this, testing each of the libraries for performance and ease of API usability would contribute in reducing development time. The following list of libraries were chosen after testing and analysis of advantages and disadvantages: NLTK for preprocessing raw text data, Gensim’s Doc2Vec implementation for feature extraction/representation, Scikit-learn for cross-validation, and NumPy for efficient vector/matrix operations as my main libraries.

**c. Doc2Vec vs. Term Frequency–Inverse Document Frequency (TF-IDF)**

Doc2Vec is a concept introduced by Tomas Mikolov and Quoc Le, which represents documents as vectors through the Distributed Memory (DM) and Distributed Bag of Words (DBOW) model. In short, DM is responsible for remembering the context (ordering of words) in a document while DBOW is responsible for predicting the context of a document. On the other hand, TF-IDF was a method of scoring the rarity of words across a collection of documents. At face-value, both options were feasible solutions for extracting features. Since speed was a strong deciding factor, tests were performed on both models which involved a training input of 5000 documents/sentences.

To avoid boilerplate code and encourage simplicity, Gensim’s implementation of Doc2Vec and Sklearn’s implementation of TF-IDF was used. The results showed that Doc2Vec took ~5 seconds to train while TF-IDF took ~10 seconds to train. Thus, Doc2Vec was chosen for the feature extraction and representation step.

**d. Splitting Training and Test Data**

Originally, a combination of the training and test data was the input into the Doc2Vec model; however, this was not the best approach for a small dataset with only 18560 records. To prevent using the test set for hyper-parameter tuning, k-fold cross-validation was the most suitable approach with the small dataset.

**III. Approach**

**a. Pre-processing**

**b. Feature Extraction and Representation**

**c. Choosing k with Cross-validation**

**d. kNN Classification**

**IV. Conclusion**

pre-processing🡪feature extraction🡪feature representation🡪cross-validation🡪classification

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 ?