report.md

```
Prakash Dhimal
George Mason University
CS 584 Theory and Applications of Data Mining
Homework 1
```

Accuracy:

```
Miner username: mistor
Accuracy as of this report: 0.64
```

Data:

Following data files were obtained from Miner2.vsnet.gmu.edu

- `1580449515_4035058_train_file.dat`
- `1580449515_4313154_test.dat`

Data pre-processing

- preserved numerical ratings (1 start, 2 star, 3 star, 4 star, and 5 star) by converting them to string values
- used `nltk.word.tokenize` to tokenize the document
- use `token.lower` to
- remove tokens in `string.punctuation`
- remove tokens in stop_words (`stopwords.words('english')`)
- remove tokens less than 3 characters long
- use `nltk.PorterStemmer()` to stemm the token

It is important to note that the order of the above list does matter. Only the training datasets is processed initially using the steps above. In addition to the steps above, training labels were seperated for use in the K-NN classification process later.

Testing data

- retrieve the raw test data.

Testing data is not processed initially. The KNN classification process is going to take a testing instance (a review) and pre-process it using the same steps above.

K-Nearest neighbor:

Text representation:

After initially spending a lot of time with TF-IDF For the representation of the text (product reviews) I implemented **TF-IDF**.

TF-IDF:

Let's define some of the terms:

```
t_{k} – term (word) k
d_{i} – document_{i}
N = total number of documents in the collection C
n_{k} = the number of documents in C that contain t_{k}
tf_{ik} = frequency of term t_{k} in document document_{i}
```

```
df_{k) = Occurrence of term k in the collection of documents
idf_{k} = log( N/df_{k} )
```

And finally:

```
TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)
```

DF

I store a list of document frequency (DF) for each word in the collection with the word as the key and document frequency (DF) as the value so that we can easily lookup the `DF` value for each word quickly and efficiently.

DF-IDF Matrix

I built the TF-IDF matrix on by looping on each document.

Distance/Similarity measure:

Cosine similarity

I used `np` module to calculate cosine similarity (normalized dot product) between two documents. For each test instance, the raw document (review) is preprocessed and it is projected into the TF-IDF matrix build in the previous step. I built a TF-IDF matrix only for the training set. The test instance is going to get a vector with respect to the TF-IDF matrix for the training set. The cosine similarity is calculated between this vector and each document vector in the TF-IDF matric (for the training set).

This cosine similarity is used to determine the nearest k - neighbors of the test instance (document) in the training data set. The documents with top `k` highest cosine similarity are selected as the neighbors for this training instances. The majority wins the label for the test instance.

This brings us to another important part in K-NN, the value of `K` . For the value of K, I used `15` . I've tried with several different values, 25 seems to yield better results.

Multiprocessing:

After tinkering with Threads and Processes in python for a long time I ended up implementing a queue to use multiple processes to find the label for each training instance. I used `multiprocessing.cpu_count() - 1` to determine the number of processes running at the same time.

Validation:

For validation:

- read the training data
- shuffle the data few times

Shuffling of the data is important because the training data has mostly positive reviews in the begining and negative reviews in the end.

Then the data is divided into training and test sets:

- 13872 data points for training
- 4625 data points for test
- rest of the data points were dropped using `dropna` for missing values

Dataset of 18497: 13872 for training 4625 for testing

`K = 15` 69.18919% accuracy

`K = 5` 67.9351% accuracy

`K = 25` 70.1189% accuracy

Following the stackoverflow answer:

One of the threads metioned "the value of k is non-parametric and a general rule of thumb in choosing the value of k is `k = sqrt(N)/2`, where N stands for the number of samples in your training dataset."

sqrt of 13872 = 117, 117/2 = 58.5, let's use 55 for the value of K. `K = 55` 68.3027% accuracy

For more details on validation, please see `src/jupyter-notebooks/validation.ipynb`

Link to the stackoverflow thread: https://stackoverflow.com/questions/11568897/value-of-k-in-k-nearest-neighbor-algorithm

Runtime:

Even with multi-threading, the run-time for this program wasn't as good as I wanted.

Validation times are reported in the validation notebook.

Runtime on a full dataset with `k=25`

```
Time taken = 4942.5809831619 Seconds
= 82.38 minutes
= 1 Hr 22.38 minutes
```

on:

```
Centos 8
Sager Laptop with 8 logical processors
```

Please see `../src/jupyter-notebook/knn_classifier_workbook.ipynb` to see a full run.