**Willy Esquivel-Lopez**

**(wesquive - G01127937)**

**(miner2 user-name): Gopher123**

**HW1 Report**

I First imported the data into a Pandas Data Frame to work with it more easily. I Began my process by cleaning the data from any noise by the provided reviews data. The approach I took was:

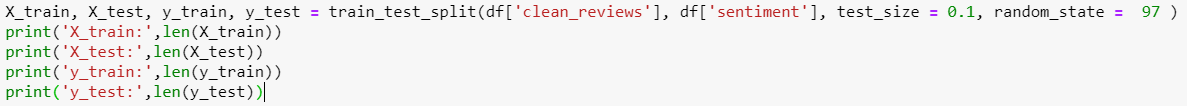
**Cleaning data process**

1. Remove all of the HTML tags – at a glance I could see several html and markup tags.
2. I also removed all punctuation also using Regex
3. I also began to eliminate stop words (such as “the”, “a”, “an”, “in”) as they are of no use
4. I also proceed to tokenize all the words so that I can use them as a bag of words and later be able to vectorize them
5. Finally, I lemmatized the words to shorten and simplify unnecessary and derived words

(such as rocks: rock, corpora: corpus, better: good)

**Splitting my training data into sets for training and testing to create my model**

With my training data containing exactly 14999 entries of data I split the values as such



This resulted in a nice split with plenty of training data and enough testing data in a 90% to 10% split to train and test my model, like so:

X\_train: 13499

X\_test: 1500

y\_train: 13499

y\_test: 1500

**Model Creation**

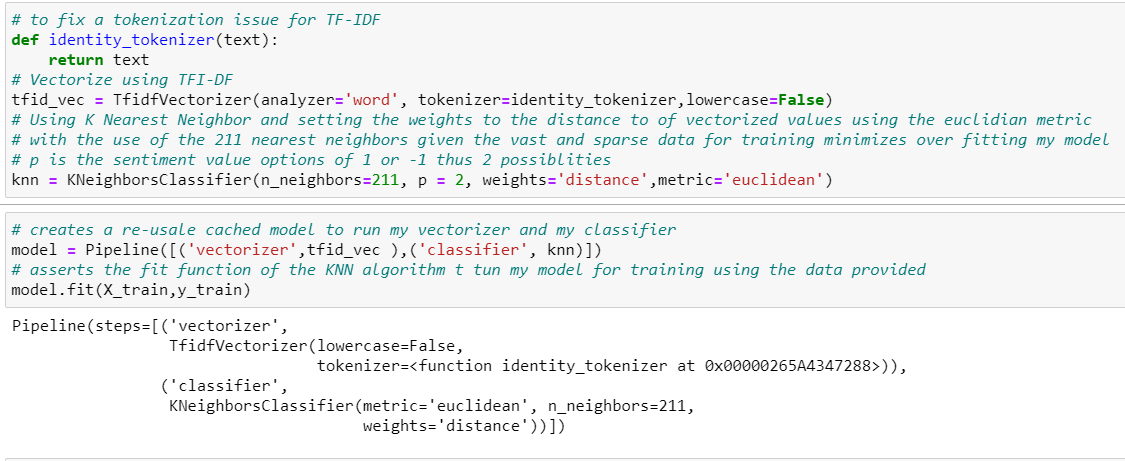
I ran into a problem with he pre tokenized values and utilizing a library which is used to make a matrix of TF-IDF features, thus I included a helper function. (this bug took some time to solve).

After using the “TfidfVectorizer” from SciKit learn, I began implementing my K-Nearest neighbor classifier algorithm using SciKit Learn :

I chose given the sparse data a high odd number of nearest neighbors = “n\_neighbors=211” since it appeared that choosing something smaller would be Over Fitting my model selection.

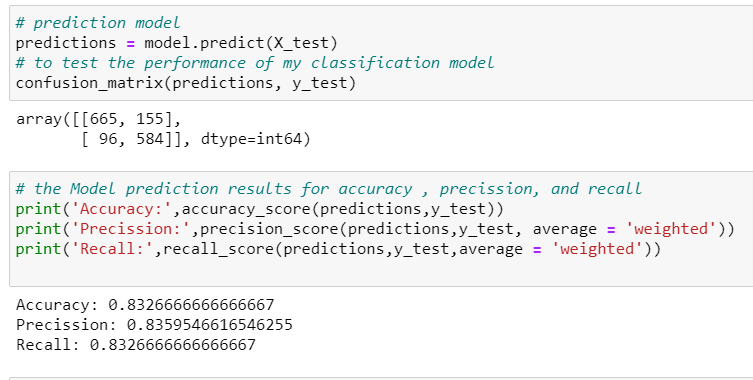
The metric I used for measuring distance was Euclidean “metric='euclidean” paired with a power metric of “p=2” since the only options in sentiment were -1 and 1.

After that I was able to cache my model using a Pipeline which is able to run both the T-FID vectorizer and my classifier. Finally, I train with my model given the parameters “X\_train,y\_train” or the IV and DV train data.

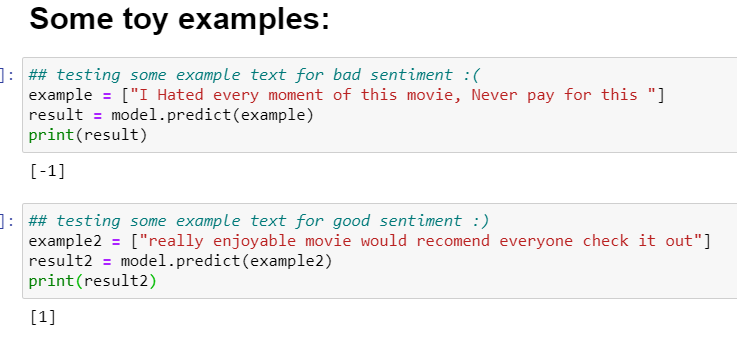


Next, I began to evaluate my classifications by doing some predictions on my split test data, I am able to gather the prediction outcome into a “confusion\_matrix”

I am then able to gauge the outcomes of my prediction on my test data with accuracy, precision, and recall scores. All these scores are taken into account and result in a positive outcome given my metrics chosen for my model:



**I preform a final but simple sanity check:**



I preform the same analysis on the Test\_data.txt and saved the results in a file “output.txt”

P.S. I could not make a good plot or graph to save my life, thus, to choose a good number of nearest neighbor and test train slip I made some iterative for and while loops and compared results and found this to produce the best outcome