**Natural Language Processing**

**Assignment #03**

**Report**

**Submitted by:**

**Karrar Hussain 13F8294**

**Purpose and Content**

This report contents the methods used for sentiment analysis for NLP assignment #03.

Naïve Bayes sentiment analysis method was the core for this assignment.

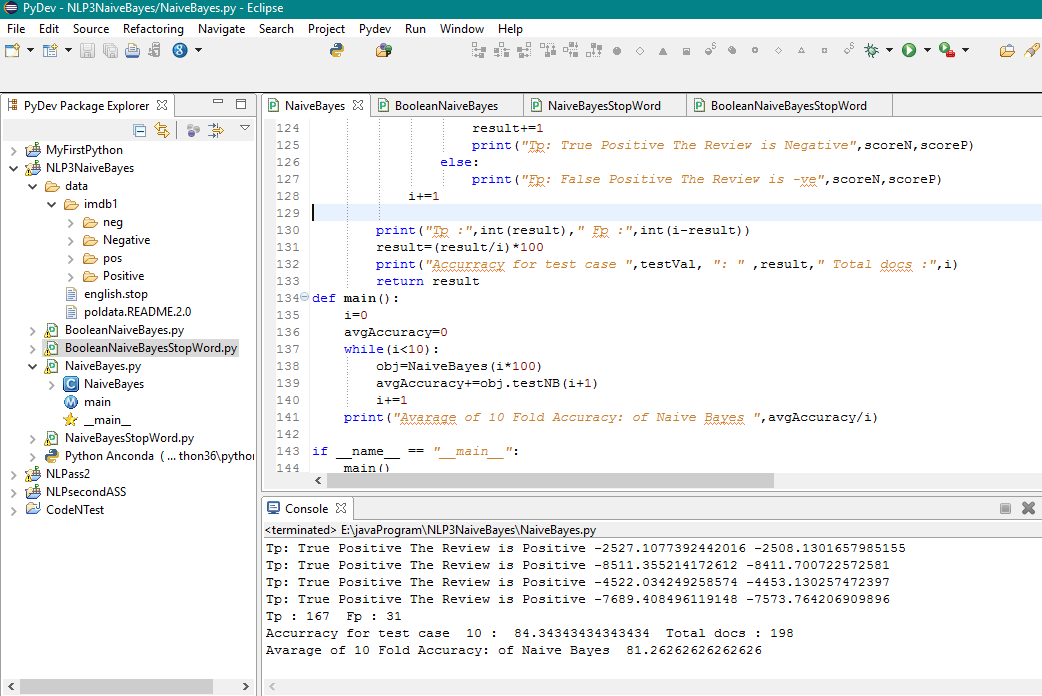
The following implementation is done in the assignment:

**Multinomial Naïve Bayes/Naïve Bayes**

I was supposed to train my Naïve Bayes for the give documents and then evaluate using 10-fold cross accuracy. The file NaiveBayes.py that is attached with the final submission contents the implementation of training of Naïve Bayes and then test. Just import the project as it is and then run NaiveBayes.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 81.26 % as shown in below screen shot.

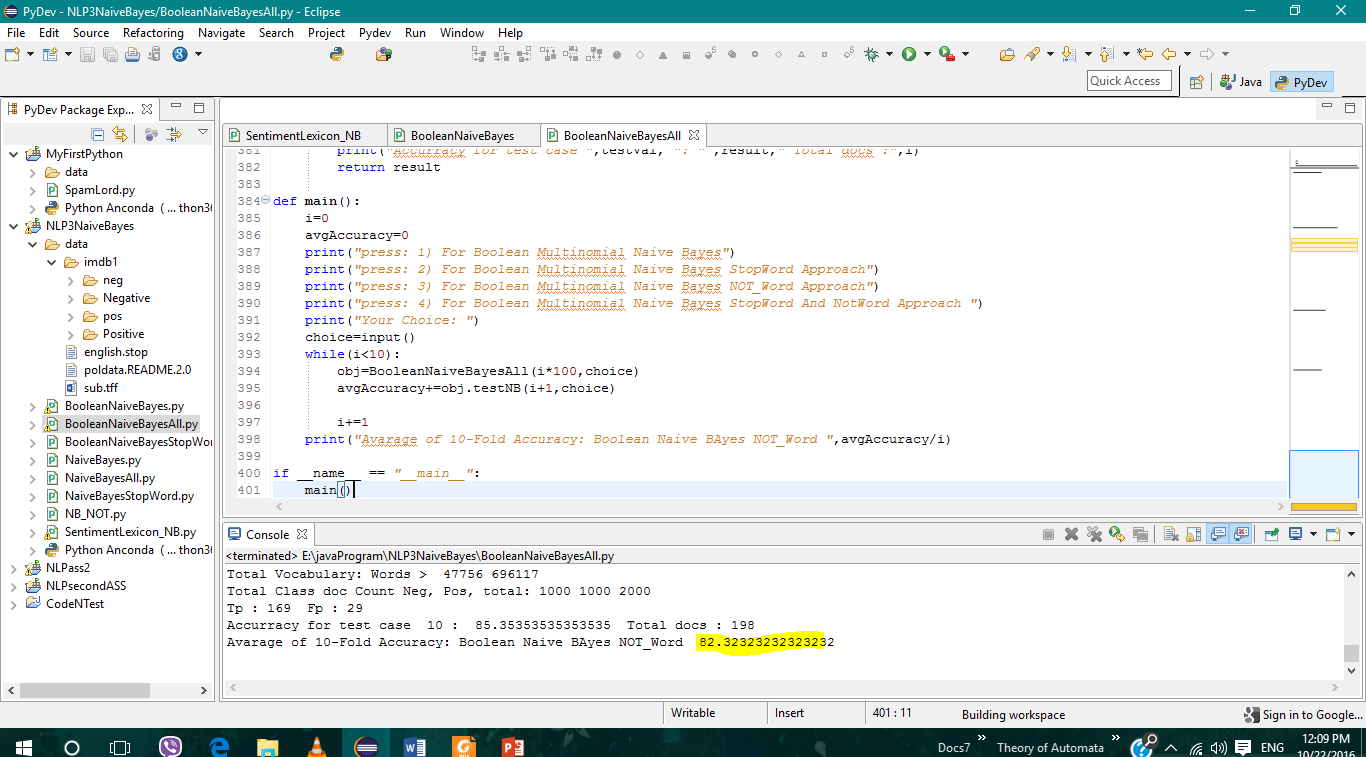
**10-fold cross test Accuracy**

The test is done on the documents taken 10% of the positive documents and 10% of negative documents while the remaining 90% is used for training of Naïve Bayes. As mentioned 10-runs are done each time the next 10% of the whole documents is taken for the testing. That’s how each time the new testing documents are taken and finally the average accuracy of 10 test cases is given as the final accuracy.



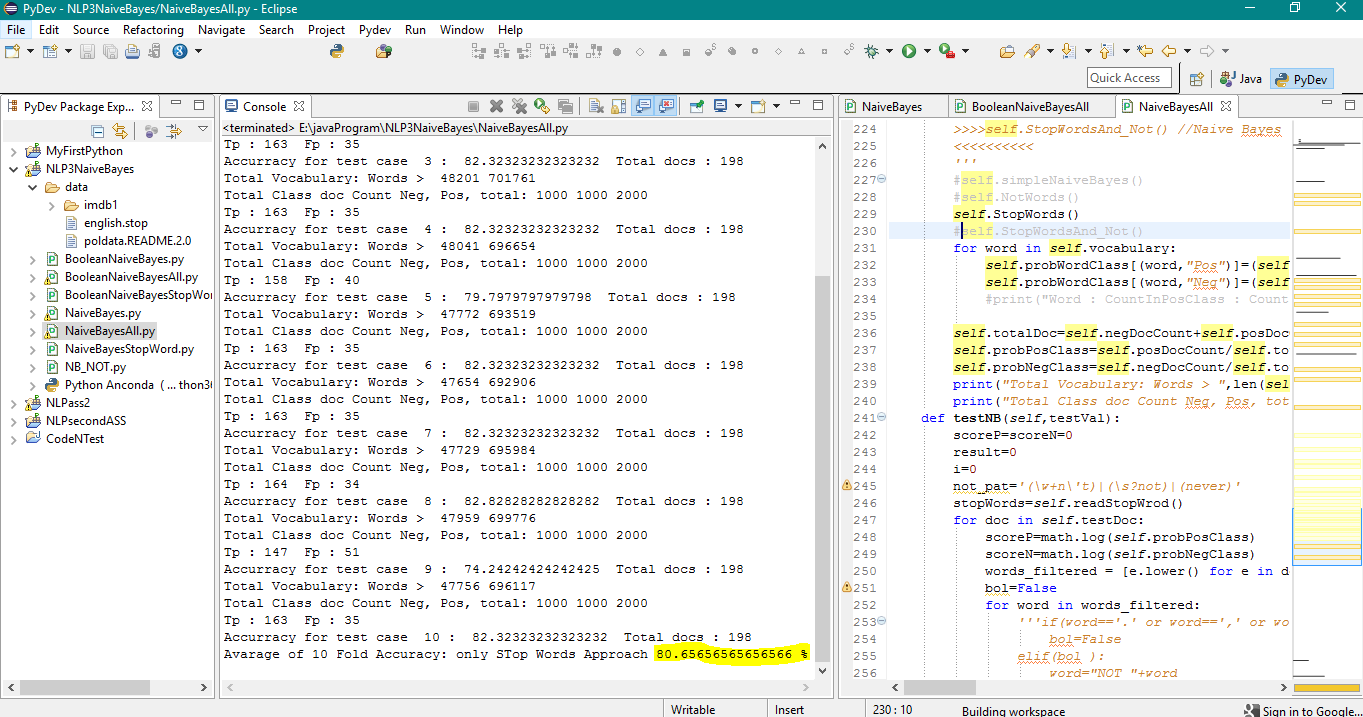
**Boolean Naïve Bayes**

I was supposed to train Boolean Naïve Bayes for the give documents (Negative and Positive reviews) and then evaluate using 10-fold cross accuracy. The file BooleanNaiveBayes.py that is attached with the final submission contains the implementation of training of Boolean Naïve Bayes and test. Just import the project as it is and then run BooleanNaiveBayes.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 82.32 % as shown in below screen shot.



**Multinomial Naïve Bayes Stop Words Approach**

I was supposed to train Naïve Bayes for the give documents (Negative and Positive reviews) neglecting stop words that was provided in English.stop file and then evaluate using 10-fold cross accuracy. The file NaiveBayesStopWords.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run NaiveBayesStopWords.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 80.6 % as shown in below screen shot.

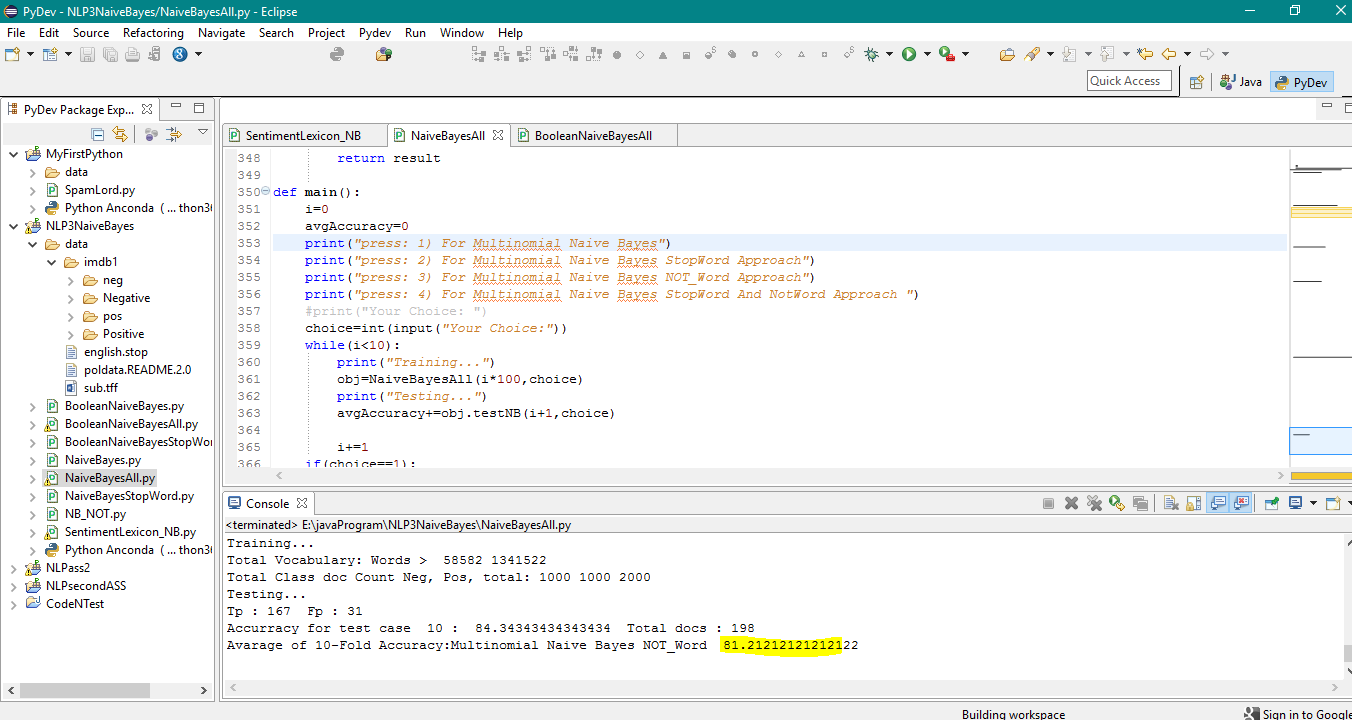


**Multinomial Naïve Bayes Negation Words Approach**

I was supposed to train Naïve Bayes for the given documents (Negative and Positive reviews) while managing the negation words (not, n’t, never) by putting NOT\_ before any word coming after negation words till next sentence. The file NB\_NOT.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run NB\_NOT.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 80.7 % as shown in below screen shot.

The Regular expression for finding negation words:

not\_pat=*'(\w+n\'t)|(\s?not)|(never)'*

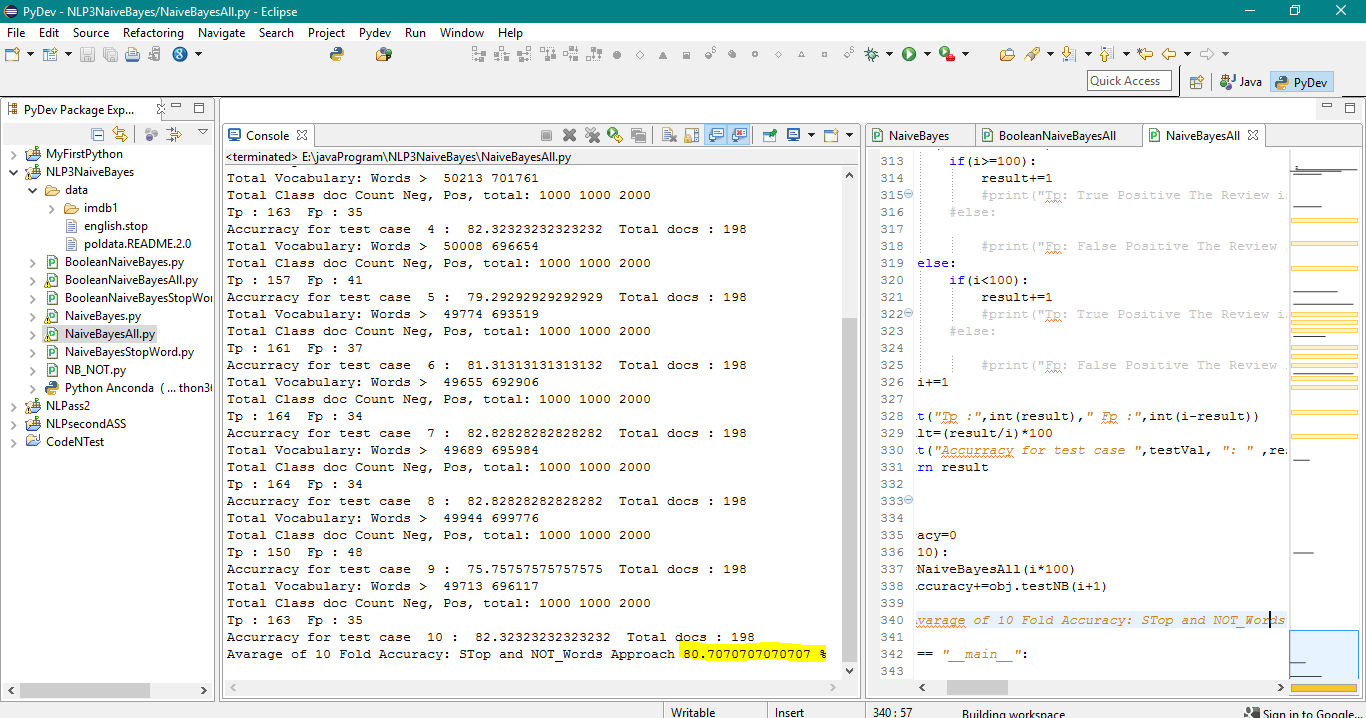


**Multinomial Naïve Bayes Stop & Negation Words Approach**

I was supposed to train Naïve Bayes for the given documents (Negative and Positive reviews) while managing the negation words (not, n’t, never) by putting NOT\_ before any word coming after negation words till next sentence and also removing stop words provide. The file NaiveBayesAll.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run NaiveBayesAll.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 80.7 % as shown in below screen shot.

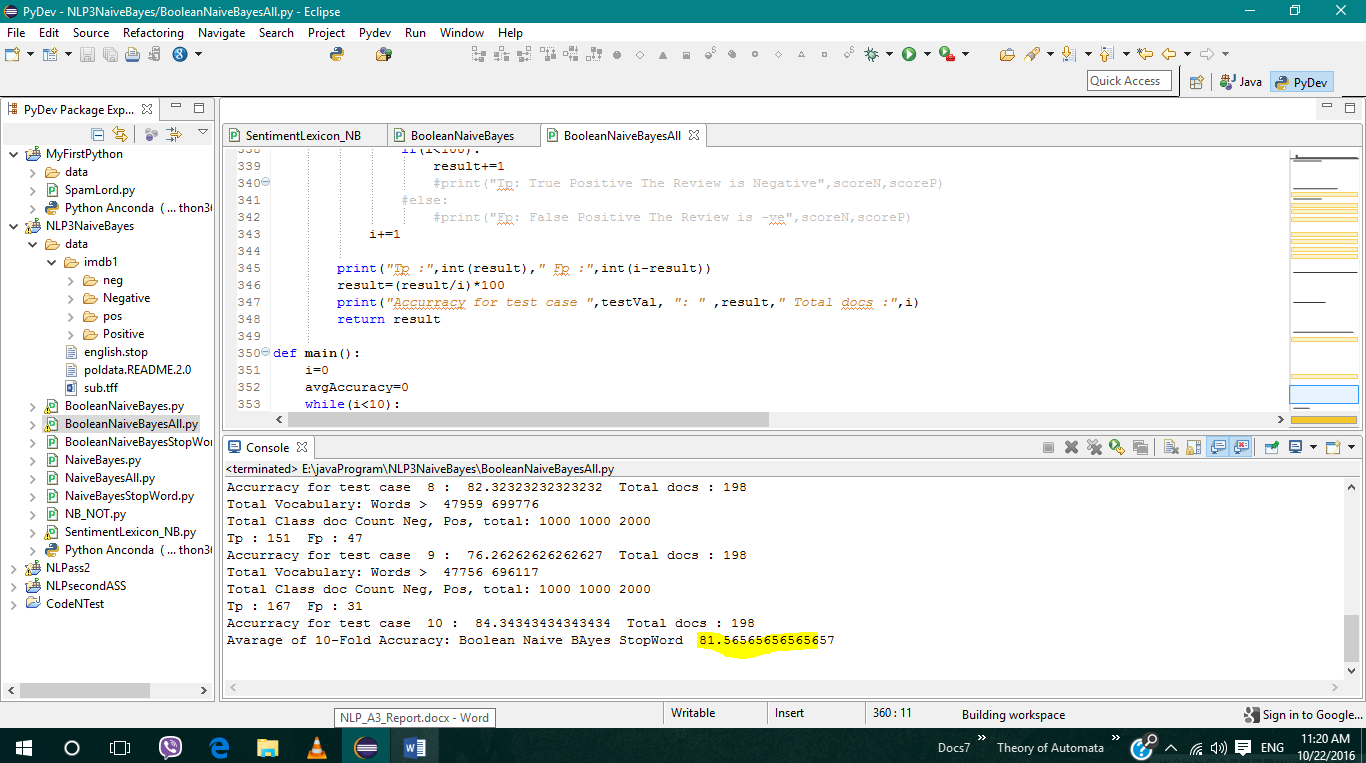
The Regular expression for finding negation words:

not\_pat=*'(\w+n\'t)|(\s?not)|(never)'*



**Boolean Naïve Bayes Stop Words Approach**

I was supposed to train Boolean Naïve Bayes for the give documents (Negative and Positive reviews) neglecting stop words that was provided in English.stop file and then evaluate using 10-fold cross accuracy. The file BooleanNaiveBayesStopWords.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run BooleanNaiveBayesStopWords.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 81.5 % as shown in below screen shot.

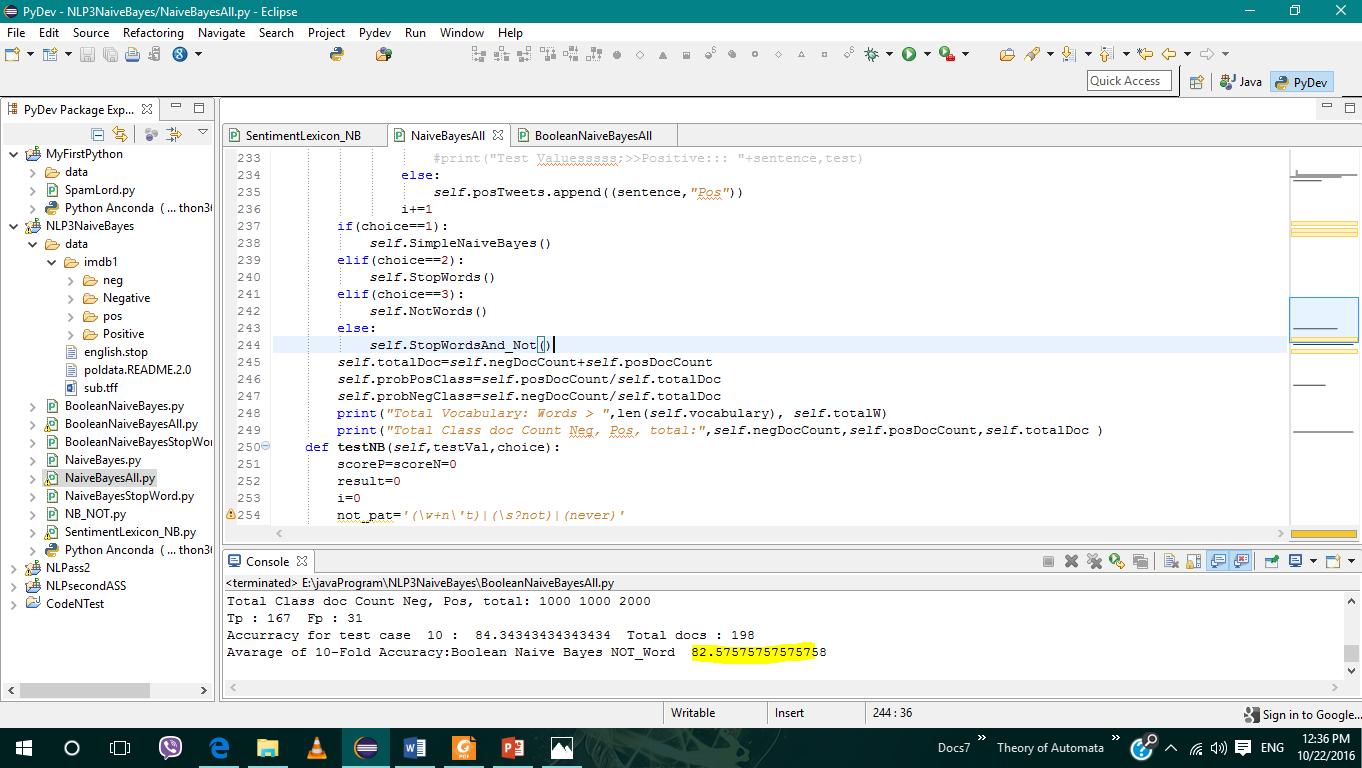


**Boolean Naïve Bayes Negation Words Approach**

I was supposed to train Naïve Bayes for the given documents (Negative and Positive reviews) while managing the negation words (not, n’t, never) by putting NOT\_ before any word coming after negation words till next sentence. The file BooleanNaiveBayesAll.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run BooleanNaiveBayesAll.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 82.57% as shown in below screen shot.

The Regular expression for finding negation words:

not\_pat=*'(\w+n\'t)|(\s?not)|(never)'*

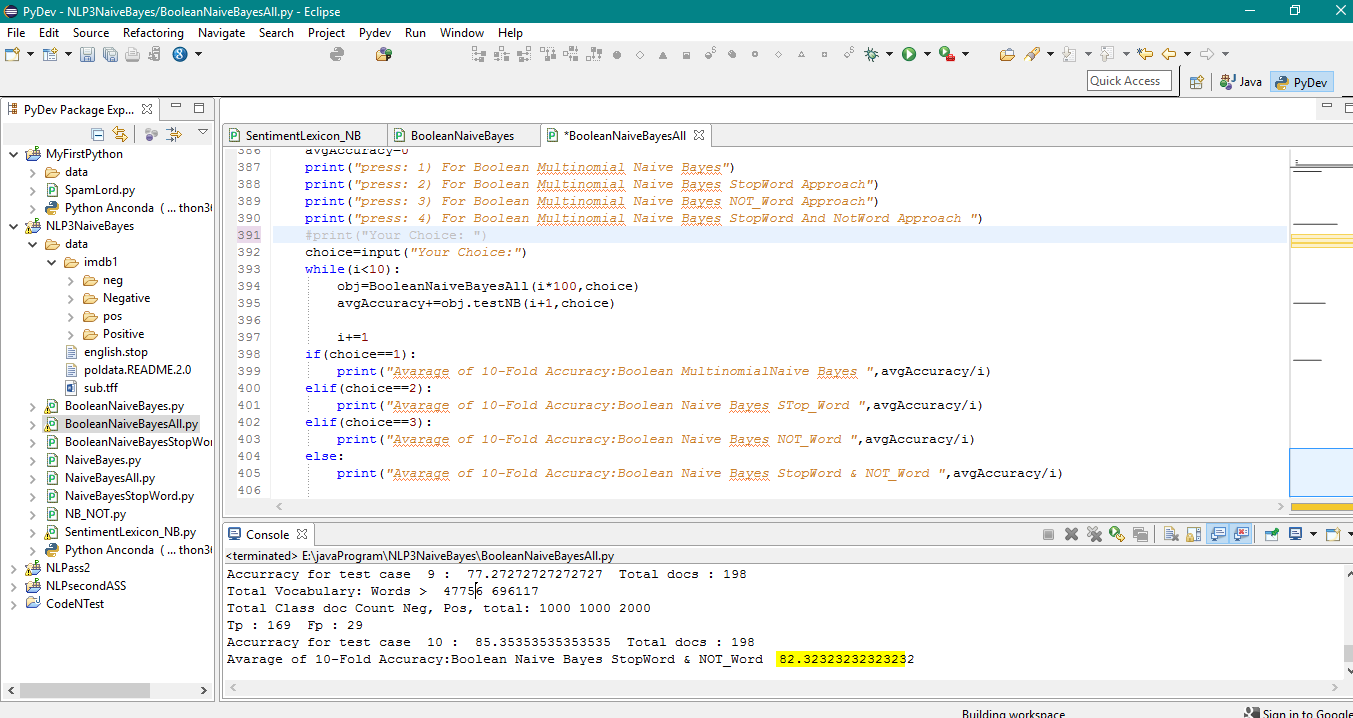


**Boolean Naïve Bayes Stop & Negation Words Approach**

I was supposed to train Naïve Bayes for the given documents (Negative and Positive reviews) while managing the negation words (not, n’t, never) by putting NOT\_ before any word coming after negation words till next sentence and also removing stop words provide. The file BooleanNaiveBayesAll.py that is attached with the final submission contains the implementation of training of Naïve Bayes and test. Just import the project as it is and then run BooleanNaiveBayesAll.py. The accuracy is taken as the average of 10-runs as were told. The accuracy for this is 79.94 % as shown in below screen shot.

The Regular expression for finding negation words:

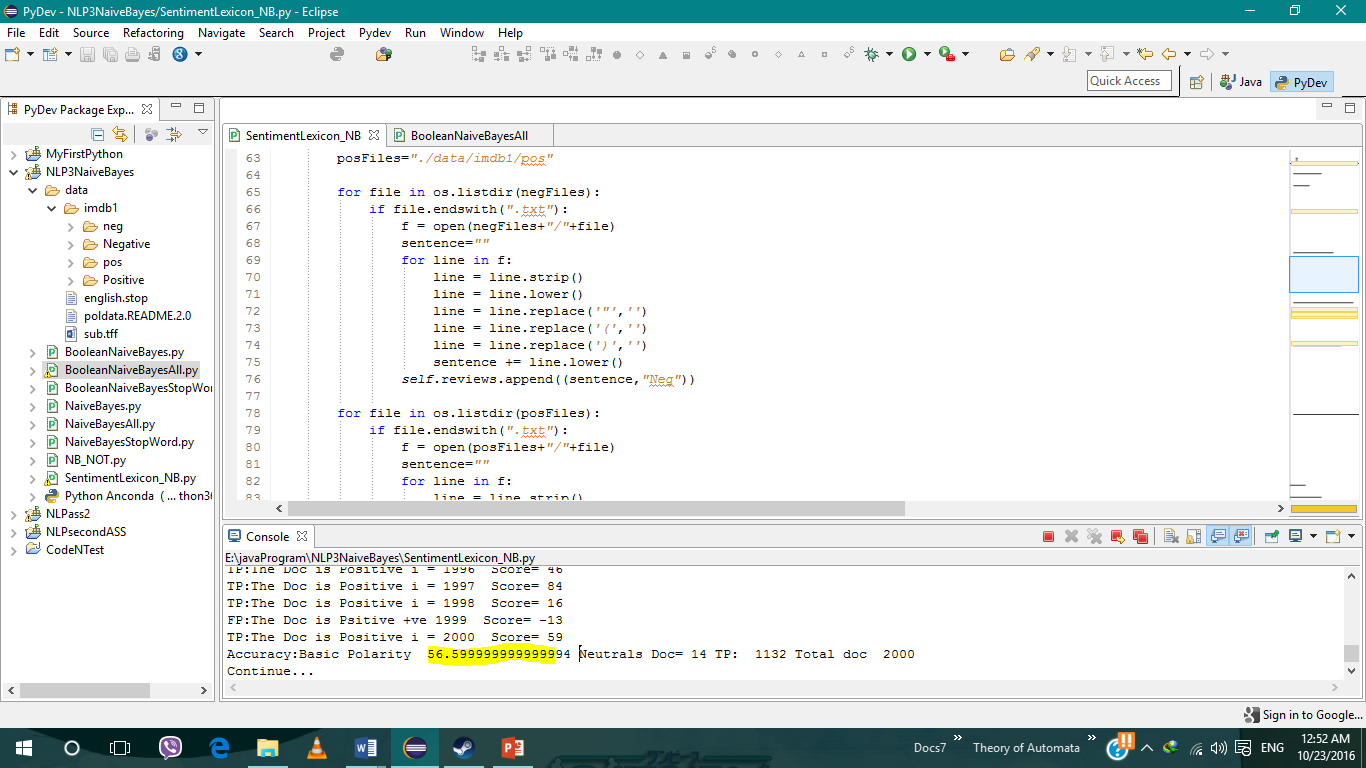
not\_pat=*'(\w+n\'t)|(\s?not)|(never)'*



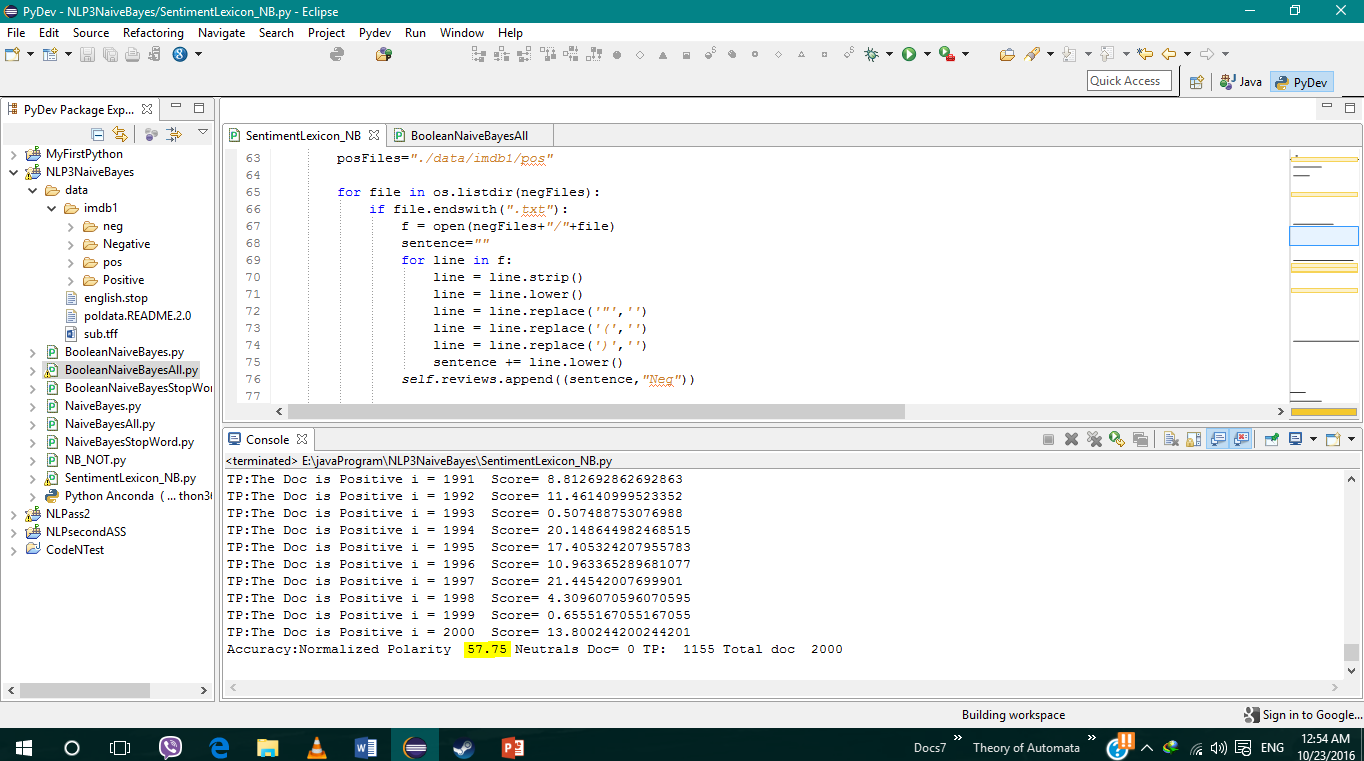
**Sentiment Lexicon Approach**

Just copy the project and sub.tff (lexicon words from MPQA) and run SentimentLexicon.py. The Basic Method give the accuracy 56.79% while Normalized gives the accuracy 57.75% as show in following screen shots:

**Basic**



**Normalized**

****

**The Best of All Above**

Accuracy table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Approach | Simple  Accuracy | Stop Word  Accuracy | Negation Words  Accuracy | Stop & Negation  Accuracy |
| Multinomial Naïve Bayes | 81.3 % | 80.65 % | 81.2 % | 80.7 % |
| Boolean Naïve Bayes | 82.32 % | 81.5 % | 82.57 % | 82.3 |
| Sentiment Lexicon | Basic 56.7%  (All documents) | Normalized 57%  (All documents) |  |  |

The best of all above is the Boolean Multinomial Naïve Bayes Negation Words Approch which give average accuracy 82.6 %.