NLP Assignment #2

Sentiment Analysis

- Sourabh S. Shenoy (UIN: 225009050)

Q1) Naive Bayes

- Regular Naive Bayes:

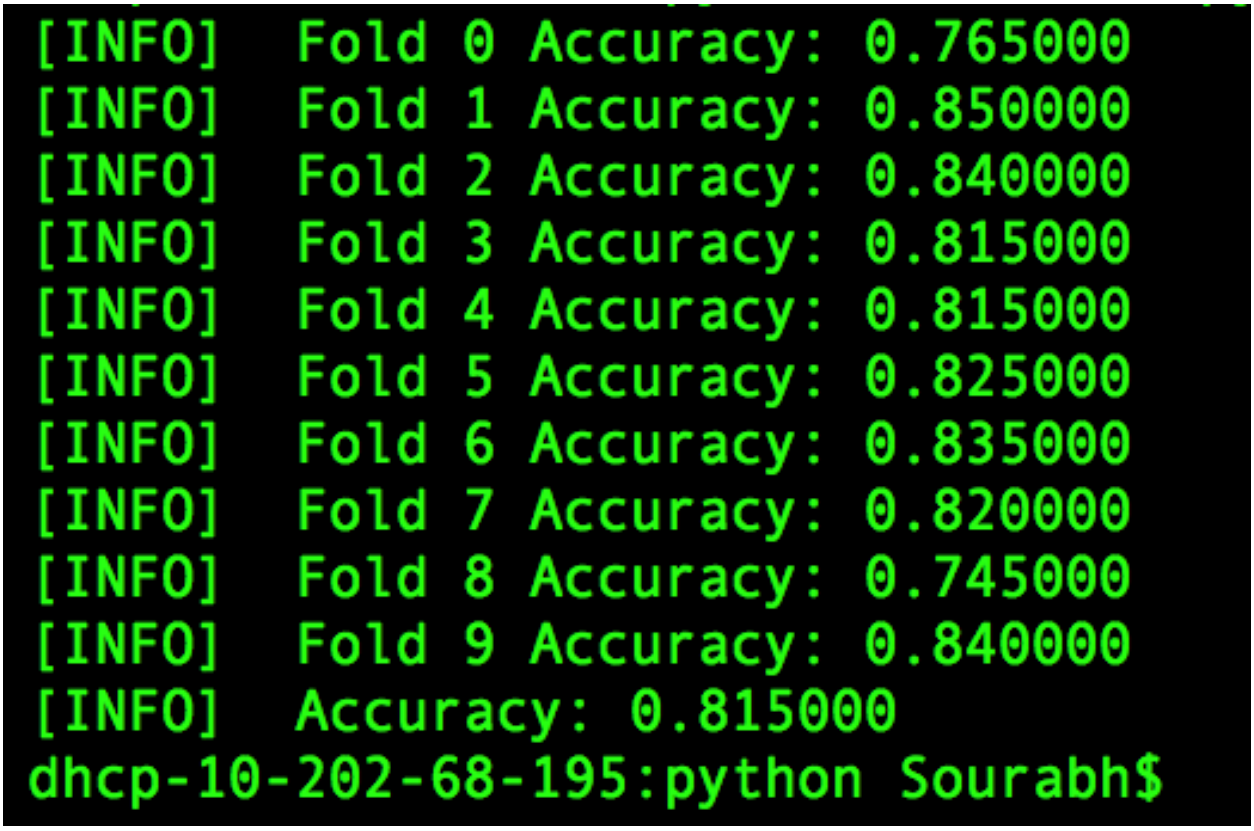
• Compilation Steps:

Open the terminal and type as shown below:

```
$ python NaiveBayes.py ../data/imdb1/
```

• Results and Analysis:

The following image shows the result obtained for regular Naive Bayes.



• Known Bugs/Limitations:

There are no known bugs for this implementation. However, the accuracy could be improved further by modeling the features better.

- Regular Naive Bayes with stop words removed:

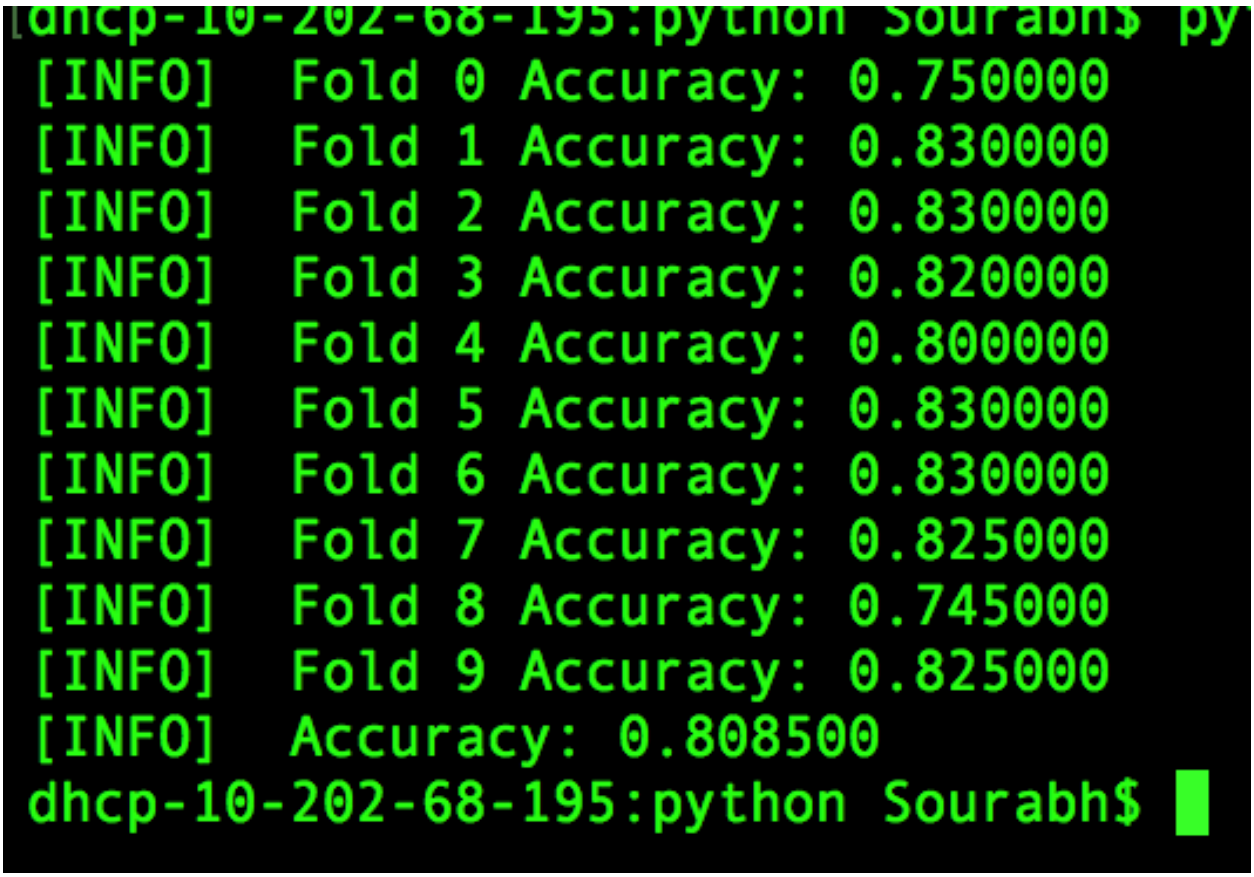
• Compilation Steps:

Open the terminal and type as shown below:

```
$ python NaiveBayes.py -f ../data/imdb1/
```

• Results and Analysis:

The following image shows the result obtained for regular Naive Bayes with stop words removed.



Without stop words, we would generally expect to see better accuracy, since they do not add meaning in any way. However, that does not seem to be the case.

- Binarized Naive Bayes:

• Compilation Steps:

Open the terminal and type as shown below:

```
h$ python NaiveBayes.py -b ../data/imdb1/
```

• Results and Analysis:

The following image shows the result obtained for Binarized Naive Bayes.

```
[INFO]  Fold 0 Accuracy: 0.795000
[INFO]  Fold 1 Accuracy: 0.840000
[INFO]  Fold 2 Accuracy: 0.840000
[INFO]  Fold 3 Accuracy: 0.825000
[INFO]  Fold 4 Accuracy: 0.835000
[INFO]  Fold 5 Accuracy: 0.830000
[INFO]  Fold 6 Accuracy: 0.840000
[INFO]  Fold 7 Accuracy: 0.845000
[INFO]  Fold 8 Accuracy: 0.785000
[INFO]  Fold 9 Accuracy: 0.855000
[INFO]  Accuracy: 0.829000
```

Binarized naive bayes is perfect for sentiment analysis, since the number of times a word appears, does not matter in determining the sentiment overall.

Q2) Perceptron

- Compilation Steps and corresponding results shown for various iterations:

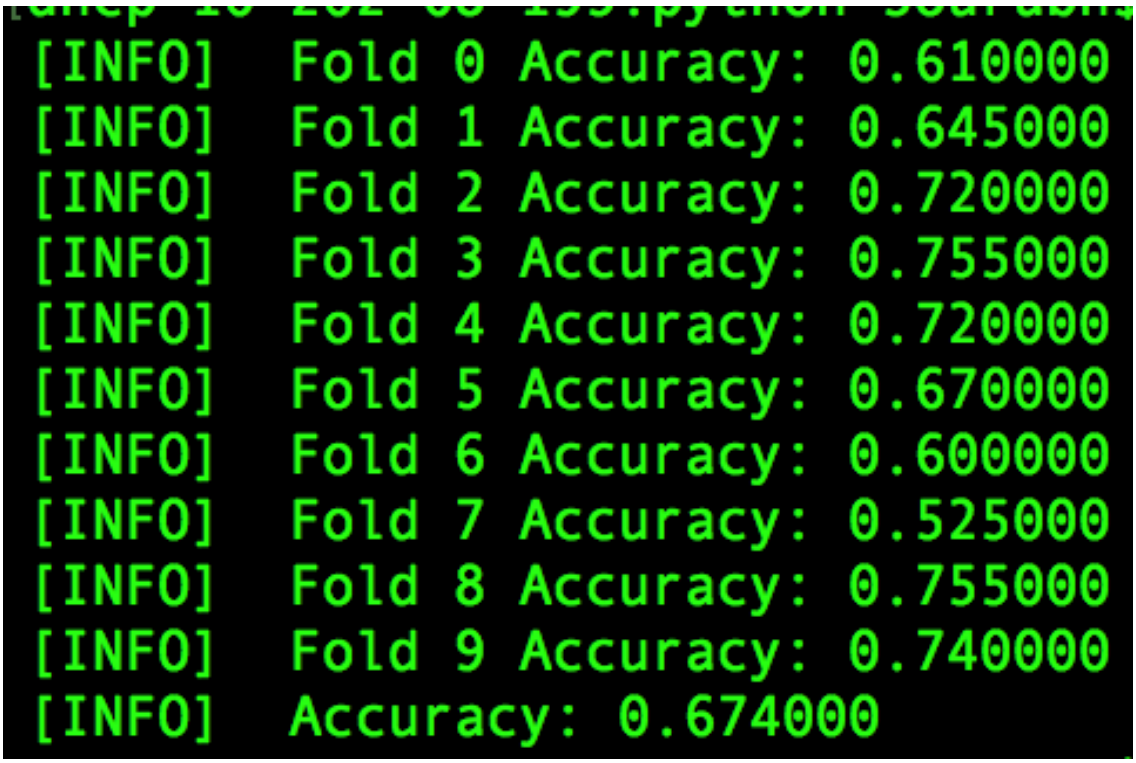
- 1 iteration:

```
python Perceptron.py ../data/imdb1/ 1
```

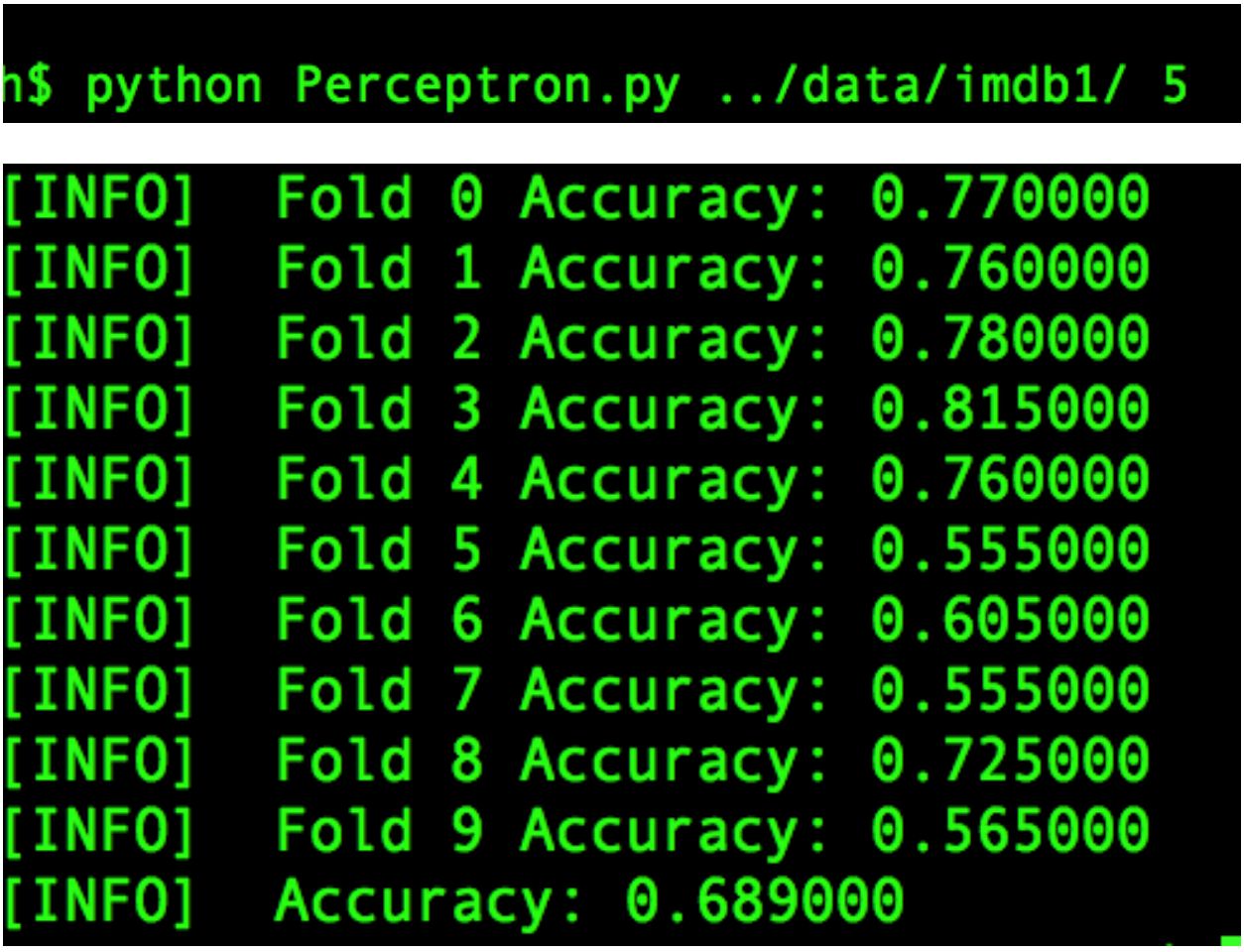
```
[INFO]  Fold 0 Accuracy: 0.695000
[INFO]  Fold 1 Accuracy: 0.660000
[INFO]  Fold 2 Accuracy: 0.715000
[INFO]  Fold 3 Accuracy: 0.780000
[INFO]  Fold 4 Accuracy: 0.600000
[INFO]  Fold 5 Accuracy: 0.710000
[INFO]  Fold 6 Accuracy: 0.520000
[INFO]  Fold 7 Accuracy: 0.525000
[INFO]  Fold 8 Accuracy: 0.735000
[INFO]  Fold 9 Accuracy: 0.515000
[INFO]  Accuracy: 0.645500
```

- 2 iterations:

```
dhcp-10-202-68-195:python Sourabh$ python Perceptron.py ../data/imdb1/ 2
```



-5 iterations:



- 10 iterations:

```
python Perceptron.py ../data/imdb1/ 10
```

```
[INFO]  Fold 0 Accuracy: 0.660000
[INFO]  Fold 1 Accuracy: 0.760000
[INFO]  Fold 2 Accuracy: 0.580000
[INFO]  Fold 3 Accuracy: 0.765000
[INFO]  Fold 4 Accuracy: 0.735000
[INFO]  Fold 5 Accuracy: 0.800000
[INFO]  Fold 6 Accuracy: 0.605000
[INFO]  Fold 7 Accuracy: 0.780000
[INFO]  Fold 8 Accuracy: 0.770000
[INFO]  Fold 9 Accuracy: 0.810000
[INFO]  Accuracy: 0.726500
```

- 20 iterations:

```
python Perceptron.py ../data/imdb1/ 20
```

```
[INFO]  Fold 0 Accuracy: 0.685000
[INFO]  Fold 1 Accuracy: 0.770000
[INFO]  Fold 2 Accuracy: 0.785000
[INFO]  Fold 3 Accuracy: 0.840000
[INFO]  Fold 4 Accuracy: 0.780000
[INFO]  Fold 5 Accuracy: 0.740000
[INFO]  Fold 6 Accuracy: 0.815000
[INFO]  Fold 7 Accuracy: 0.775000
[INFO]  Fold 8 Accuracy: 0.755000
[INFO]  Fold 9 Accuracy: 0.755000
[INFO]  Accuracy: 0.770000
```

- 100 iterations:

```
$ python Perceptron.py ../data/imdb1/ 100
```

```
[INFO]  Fold 0 Accuracy: 0.810000
[INFO]  Fold 1 Accuracy: 0.765000
[INFO]  Fold 2 Accuracy: 0.765000
[INFO]  Fold 3 Accuracy: 0.805000
[INFO]  Fold 4 Accuracy: 0.780000
[INFO]  Fold 5 Accuracy: 0.835000
[INFO]  Fold 6 Accuracy: 0.795000
[INFO]  Fold 7 Accuracy: 0.725000
[INFO]  Fold 8 Accuracy: 0.760000
[INFO]  Fold 9 Accuracy: 0.805000
[INFO]  Accuracy: 0.784500
```

We can see that the accuracy improves with the number of iterations. Supposedly, if perceptron algorithm is run for a large number of times, then we get fairly accurate output.

- Some pointers about the code:

The “engwords-xxxx.txt” files are used as features to determine the initial weights. These are taken from various sources online, and contain different number of words (mentioned in the filename). If we desire better accuracy, then we can use the file which has more words. However, this also results in a significant increase in the time taken for the execution of the program. For the results shown here, “engwords-5000.txt” file was chosen.

- Known Bugs/Limitations:

Takes a lot of time to execute, and the time taken rises with the number of iterations as well as the number of words chosen as features.