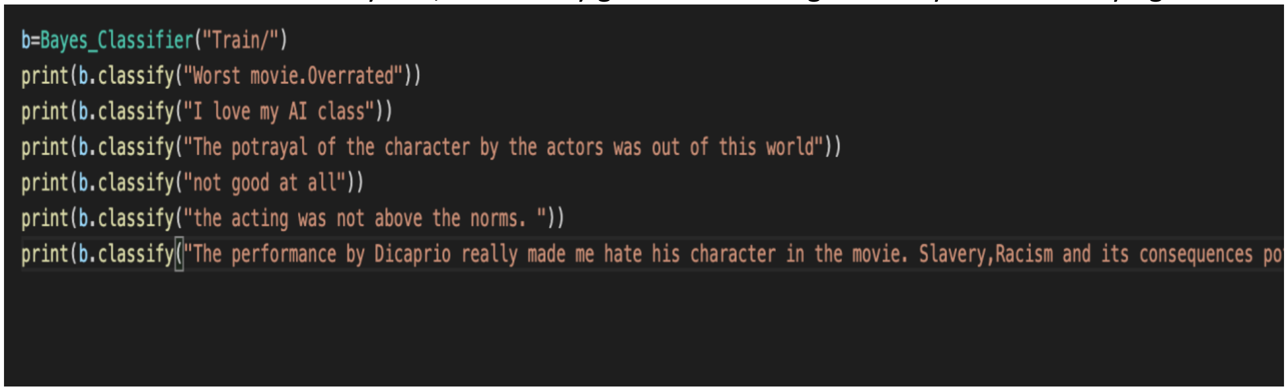
Introduction to Artificial Intelligence – CS 510
Machine Learning Programming Assignment – Sentiment Analysis

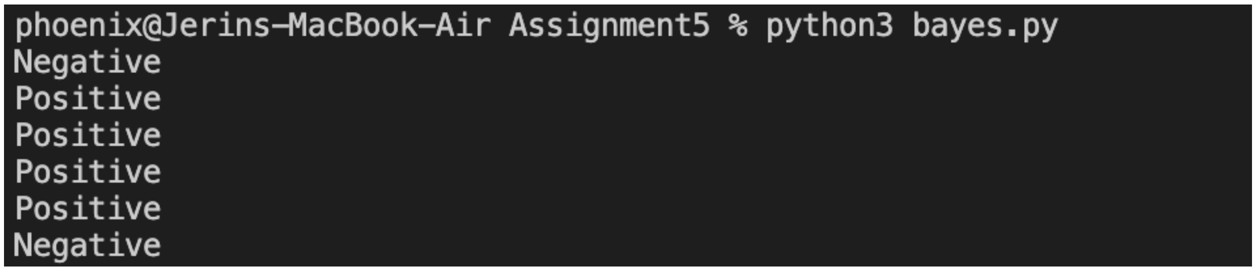
Naive Bayes Classifier:

The naïve bayes classifier just tokenizes the words and only unigrams are considered as features.
It doesn't take into consideration the length of the document, or amount of capitalization or punctuation used.
Reflection Question:
As given in the test below, the first 3 statements are classified correctly.
The last 3 statements are classified incorrectly.
This is mainly because of the following reasons:

- "not good"," not above the norms" => bad reviews. But the classifier doesn't know this as it was trained using unigrams only. If bigrams are used for training this can be avoided.
- Some words like slavery, racism etc. can appear more in negative documents. So if a movie is described by this, it normally goes into the negative tray when classifying.



```
b=Bayes_Classifier("Train/")
print(b.classify("Worst movie.Overrated"))
print(b.classify("I love my AI class"))
print(b.classify("The potrayal of the character by the actors was out of this world"))
print(b.classify("not good at all"))
print(b.classify("the acting was not above the norms. "))
print(b.classify("The performance by Dicaprio really made me hate his character in the movie. Slavery,Racism and its consequences po
```



```
phoenix@Jerins-MacBook-Air Assignment5 % python3 bayes.py
Negative
Positive
Positive
Positive
Positive
Negative
```

Evaluation:

The formulas used for calculation were:

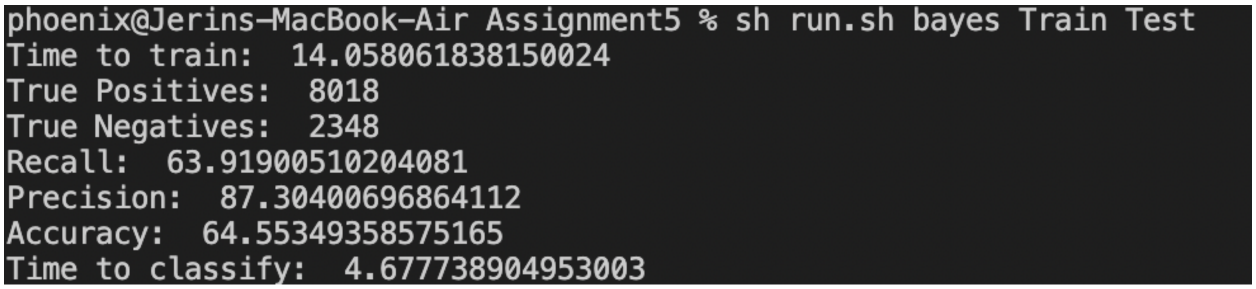
Accuracy= (TP+TN)/(TP+TN+FP+FN)
Recall=TP/(TP+FN)
Precision= TP/(TP+FP)

The below evaluation was carried out for Train file size of 50000 files and test file of size 16000 files. I made sure that all files were distinct in both folders. The train file was divided in a ration of 55:45 between positive and negative files.

To run:

```
sh run.sh [Classifier name] [Train directory] [Test directory]
```

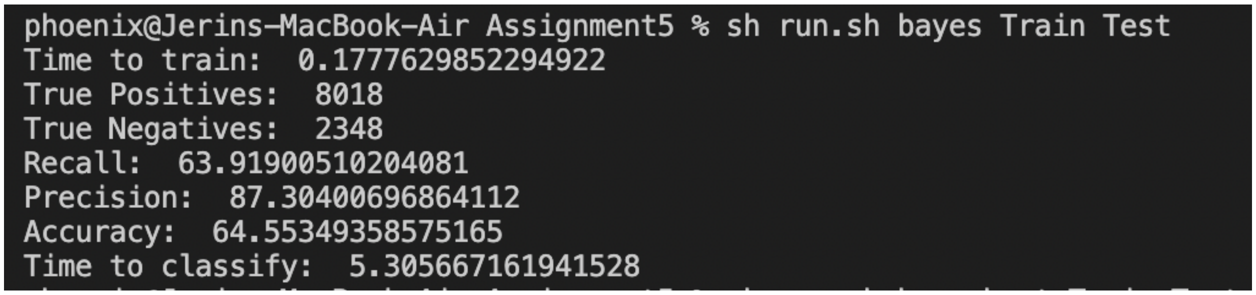
Eg: sh run.sh bayes Train Test



```
phoenix@Jerins-MacBook-Air Assignment5 % sh run.sh bayes Train Test
Time to train:  14.058061838150024
True Positives:  8018
True Negatives:  2348
Recall:  63.91900510204081
Precision:  87.30400696864112
Accuracy:  64.55349358575165
Time to classify:  4.677738904953003
```

If pickled files exist, classification happens directly. To retrain, delete files created, namely positive_reviews, negative_reviews.

Once Trained:



```
phoenix@Jerins-MacBook-Air Assignment5 % sh run.sh bayes Train Test
Time to train:  0.1777629852294922
True Positives:  8018
True Negatives:  2348
Recall:  63.91900510204081
Precision:  87.30400696864112
Accuracy:  64.55349358575165
Time to classify:  5.305667161941528
```

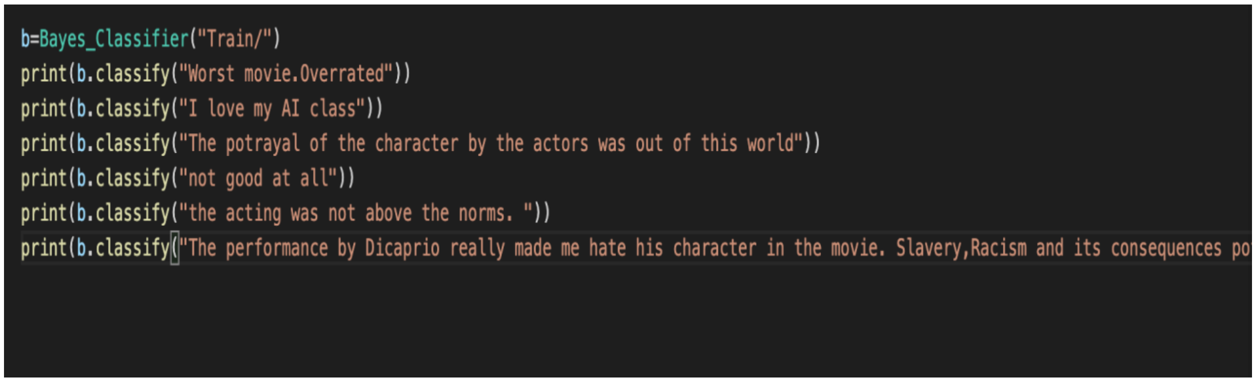
Best Bayes Classifier:

In addition to unigrams, the best bayes classifier makes use of the following additional features for training:

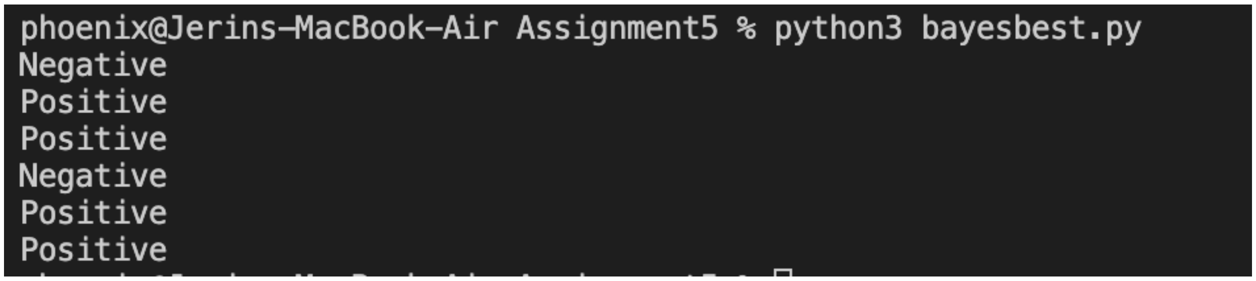
- Bigrams: Two words considered at a time.
- File Length: Length of sentence in each file
- Amount of capitalization: Capitalized words are considered separately while training
- Punctuation: Punctuations obtained from tokenized list are considered for training.

Reflection Question:
As given in the test below, the first 4 and last statements are classified correctly.
This is mainly because of the following reasons:

- “not good” => bad reviews. The classifier knows this as its trained using bigrams as well. Hence its classified correctly.



```
b=Bayes_Classifier("Train/")
print(b.classify("Worst movie.Overrated"))
print(b.classify("I love my AI class"))
print(b.classify("The potrayal of the character by the actors was out of this world"))
print(b.classify("not good at all"))
print(b.classify("the acting was not above the norms. "))
print(b.classify("The performance by Dicaprio really made me hate his character in the movie. Slavery,Racism and its consequences po
```



```
phoenix@Jerins-MacBook-Air Assignment5 % python3 bayesbest.py
Negative
Positive
Positive
Negative
Positive
Positive
```

Evaluation:

The formulas used for calculation were:

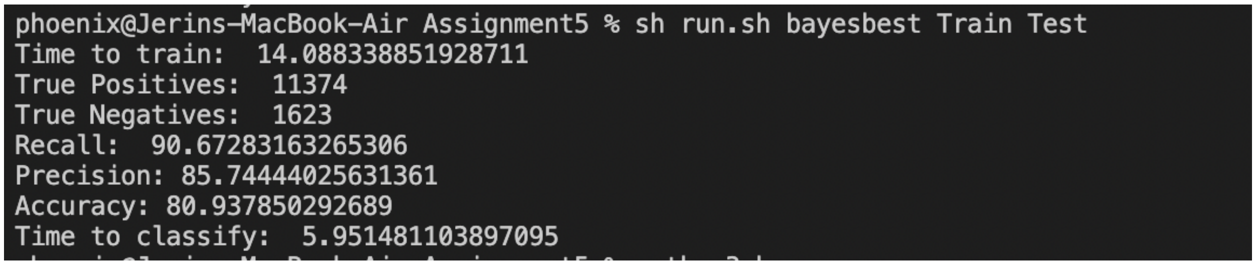
Accuracy= (TP+TN)/(TP+TN+FP+FN)
Recall=TP/(TP+FN)
Precision= TP/(TP+FP)

The below evaluation was carried out for Train file size of 50000 files and test file of size 16000 files. I made sure that all files were distinct in both folders. The train file was divided in a ration of 55:45 between positive and negative files.

To run:

```
sh run.sh [Classifier name] [Train directory] [Test directory]
```

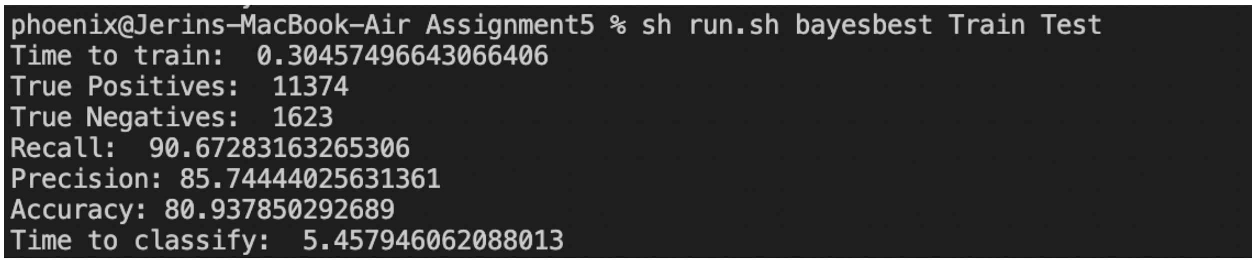
Eg: sh run.sh bayesbest Train Test



```
phoenix@Jerins-MacBook-Air Assignment5 % sh run.sh bayesbest Train Test
Time to train:  14.088338851928711
True Positives:  11374
True Negatives:  1623
Recall:  90.67283163265306
Precision: 85.74444025631361
Accuracy: 80.937850292689
Time to classify:  5.951481103897095
```

If pickled files exist, classification happens directly. To retrain, delete files created, namely positive_reviews_best, negative_reviews_best.

Once trained:



```
phoenix@Jerins-MacBook-Air Assignment5 % sh run.sh bayesbest Train Test
Time to train:  0.30457496643066406
True Positives:  11374
True Negatives:  1623
Recall:  90.67283163265306
Precision: 85.74444025631361
Accuracy: 80.937850292689
Time to classify:  5.457946062088013
```

- **As seen here, the accuracy for bayesbest (81%) is improved by a whopping 16% when compared to the previous naïve bayes approach(65%).**
- **Recall improves by 28% for bayesbest.**
- **Precision however decreases by 2% for bayesbest.**