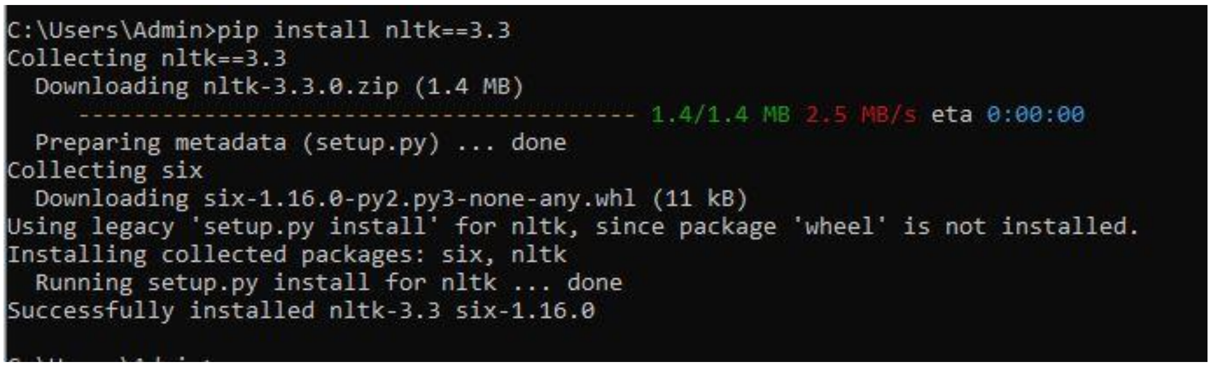
SNA MINI PROJECT

Aim: Sentiment Analysis

Code:

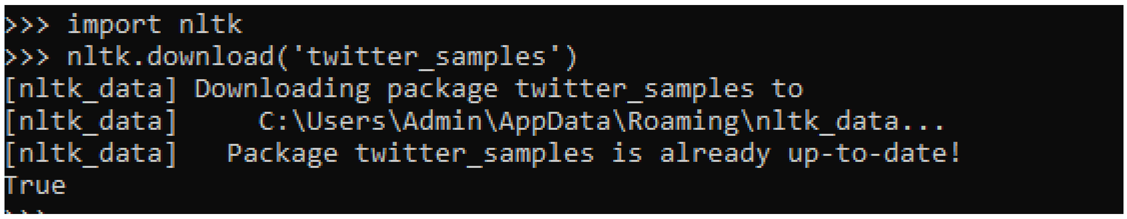
I am going to install nltk package for this.We will do this by using command:

pip install nltk==3.3



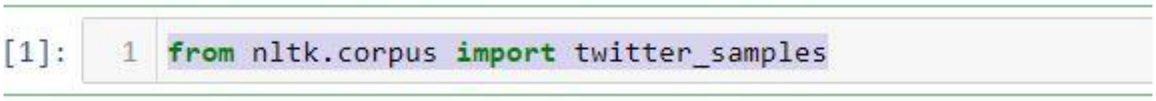
```
C:\Users\Admin>pip install nltk==3.3
Collecting nltk==3.3
  Downloading nltk-3.3.0.zip (1.4 MB)
     ---------------------------------------- 1.4/1.4 MB 2.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Using legacy 'setup.py install' for nltk, since package 'wheel' is not installed.
Installing collected packages: six, nltk
  Running setup.py install for nltk ... done
Successfully installed nltk-3.3 six-1.16.0
```

Now as nltk package is installed successfully. We will proceed to download twitter samples.



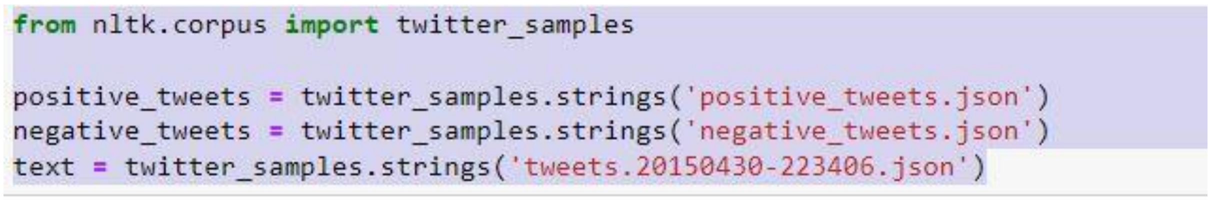
```
>>> import nltk
>>> nltk.download('twitter_samples')
[nltk_data] Downloading package twitter_samples to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package twitter_samples is already up-to-date!
True
```

This will import 3 datasets from NLTK.



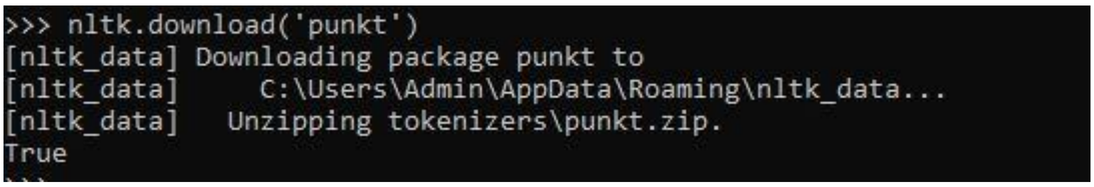
```
[1]:  from nltk.corpus import twitter_samples
```

Next we will be creating variables for positive , negative tweets and text where strings() method will print all the tweets as strings.



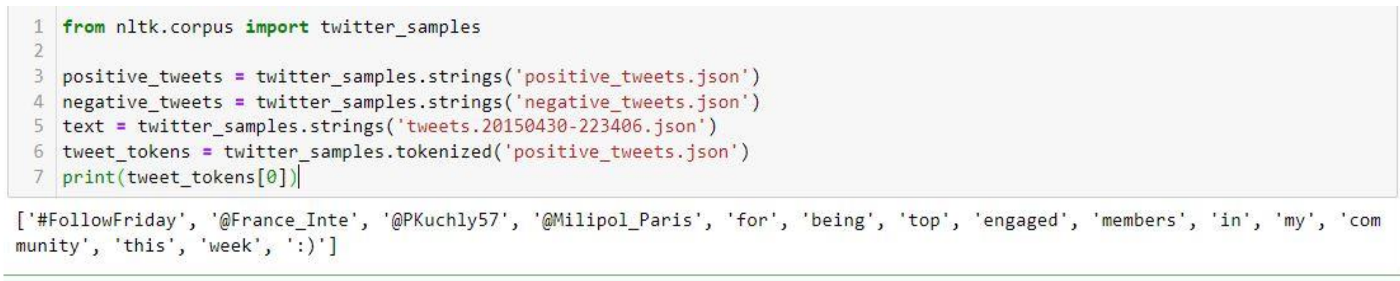
```
from nltk.corpus import twitter_samples

positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
```

Now we will be downloading resource punkt which is a pre-trained model that helps to tokenize words and sentences.



```
>>> nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
True
```

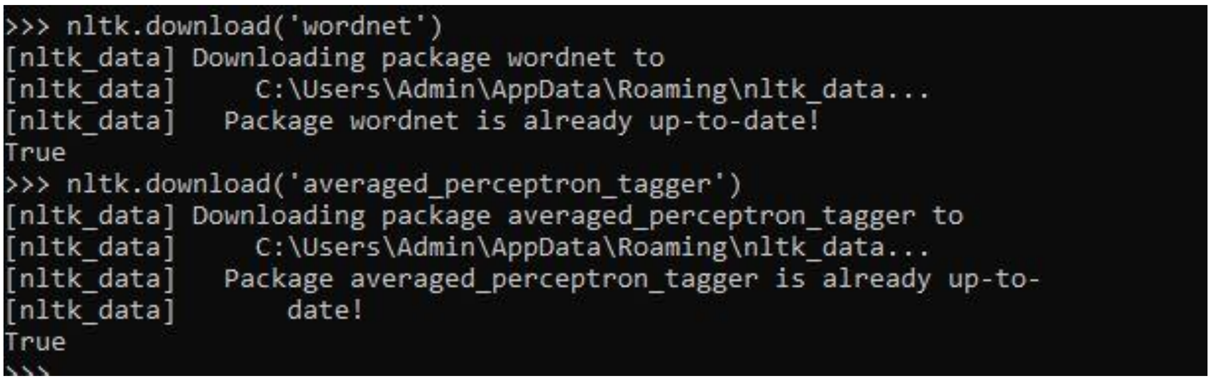
We will create an object that tokenizes positive_tweets.json dataset. The tokenized() method will return special characters in this case @, _ .



```
from nltk.corpus import twitter_samples

positive_tweets = twitter_samples.strings('positive_tweets.json')
negative_tweets = twitter_samples.strings('negative_tweets.json')
text = twitter_samples.strings('tweets.20150430-223406.json')
tweet_tokens = twitter_samples.tokenized('positive_tweets.json')
print(tweet_tokens[0])
```

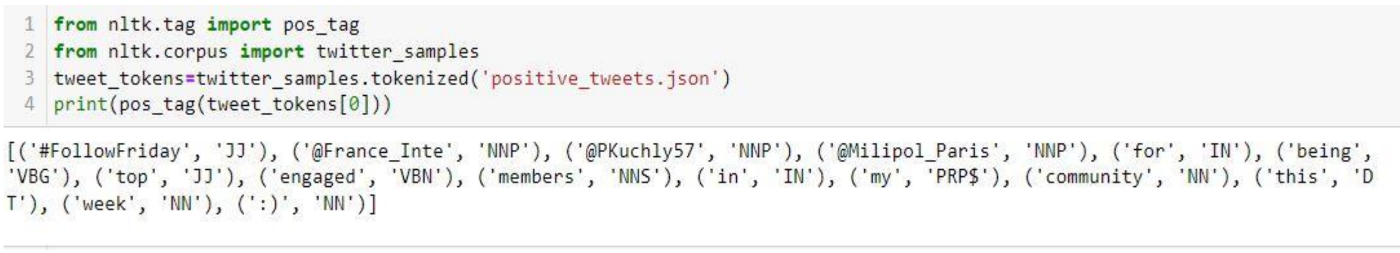
```
['#FollowFriday', '@France_Inte', '@PKuchly57', '@Milipol_Paris', 'for', 'being', 'top', 'engaged', 'members', 'in', 'my', 'com
munity', 'this', 'week', ':)']
```

Now we will be adding another resource: ‘wordnet’ and ‘average_perceptron_tagger’



```
>>> nltk.download('wordnet')
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
>>> nltk.download('averaged_perceptron_tagger')
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
>>>
```

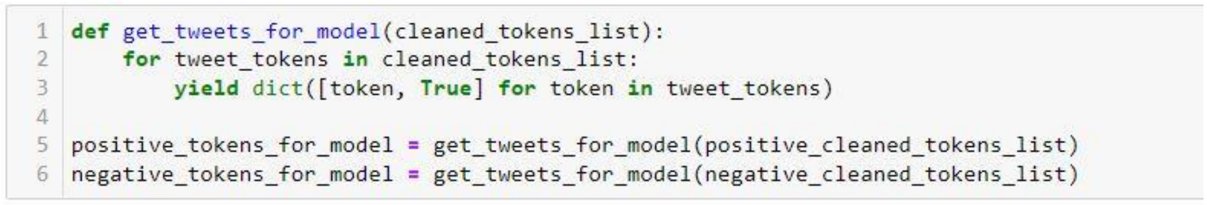
Now we will determine context for each word in our text. In python pos_tag function is used for that.



```python
from nltk.tag import pos_tag
from nltk.corpus import twitter_samples
tweet_tokens=twitter_samples.tokenized('positive_tweets.json')
print(pos_tag(tweet_tokens[0]))
```

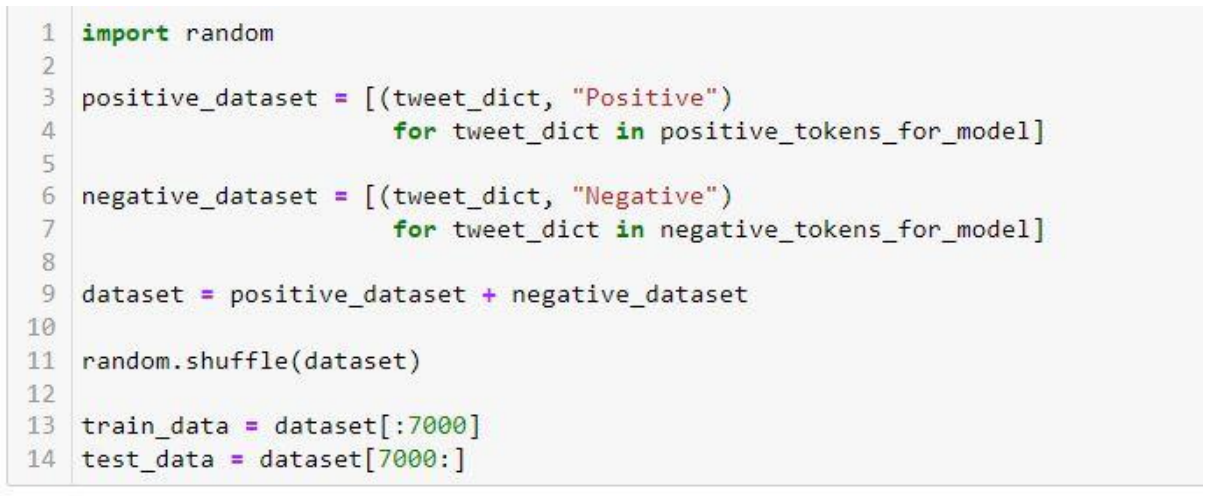
```
[('#FollowFriday', 'JJ'), ('@France_Inte', 'NNP'), ('@PKuchly57', 'NNP'), ('@Milipol_Paris', 'NNP'), ('for', 'IN'), ('being',
'VBG'), ('top', 'JJ'), ('engaged', 'VBN'), ('members', 'NNS'), ('in', 'IN'), ('my', 'PRP$'), ('community', 'NN'), ('this', 'D
T'), ('week', 'NN'), (':)', 'NN')]
```

We will be creating dictionaries for tweets.



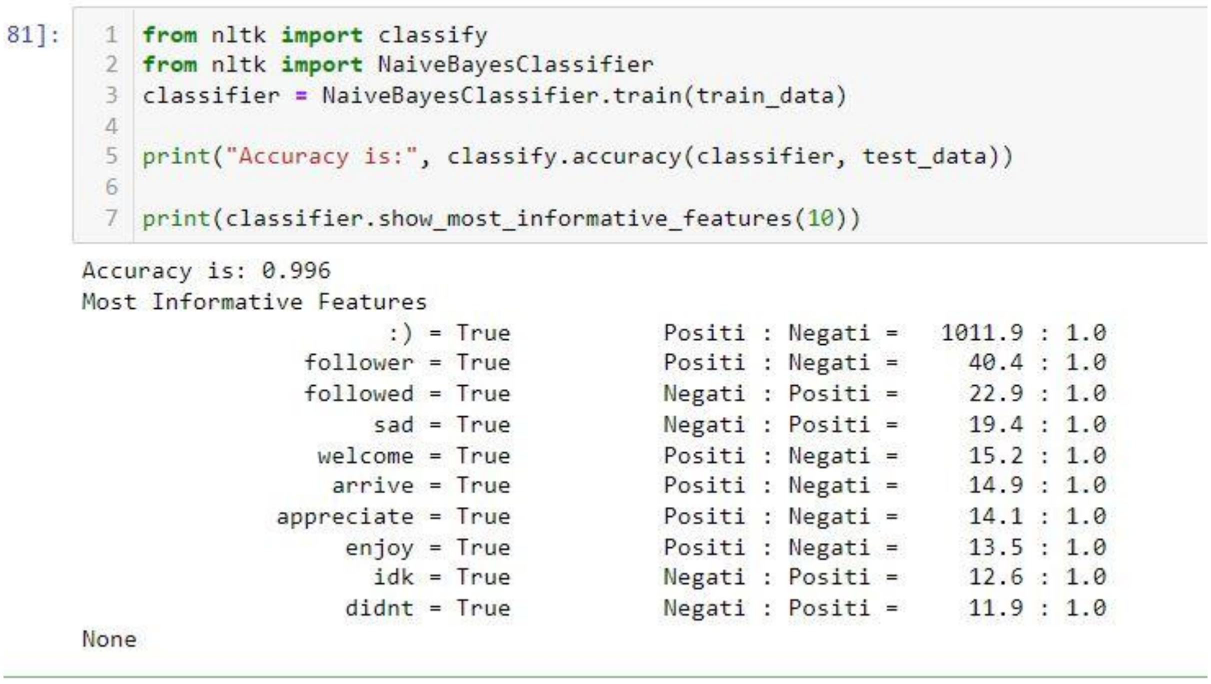
```python
def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
        yield dict([token, True] for token in tweet_tokens)

positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)
```

This code attaches positive or negative label to each tweet.



```python
import random

positive_dataset = [(tweet_dict, "Positive")
                     for tweet_dict in positive_tokens_for_model]

negative_dataset = [(tweet_dict, "Negative")
                     for tweet_dict in negative_tokens_for_model]

dataset = positive_dataset + negative_dataset

random.shuffle(dataset)

train_data = dataset[:7000]
test_data = dataset[7000:]
```

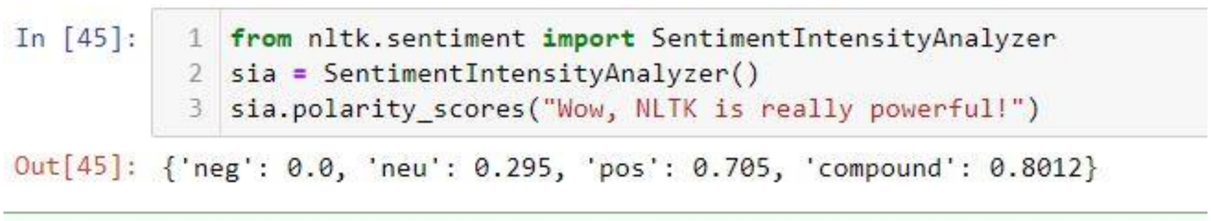
We will create a model by using NaiveBayesClassifier and will be testing it by accuracy() method.



```python
81]:
from nltk import classify
from nltk import NaiveBayesClassifier
classifier = NaiveBayesClassifier.train(train_data)

print("Accuracy is:", classify.accuracy(classifier, test_data))

print(classifier.show_most_informative_features(10))
```

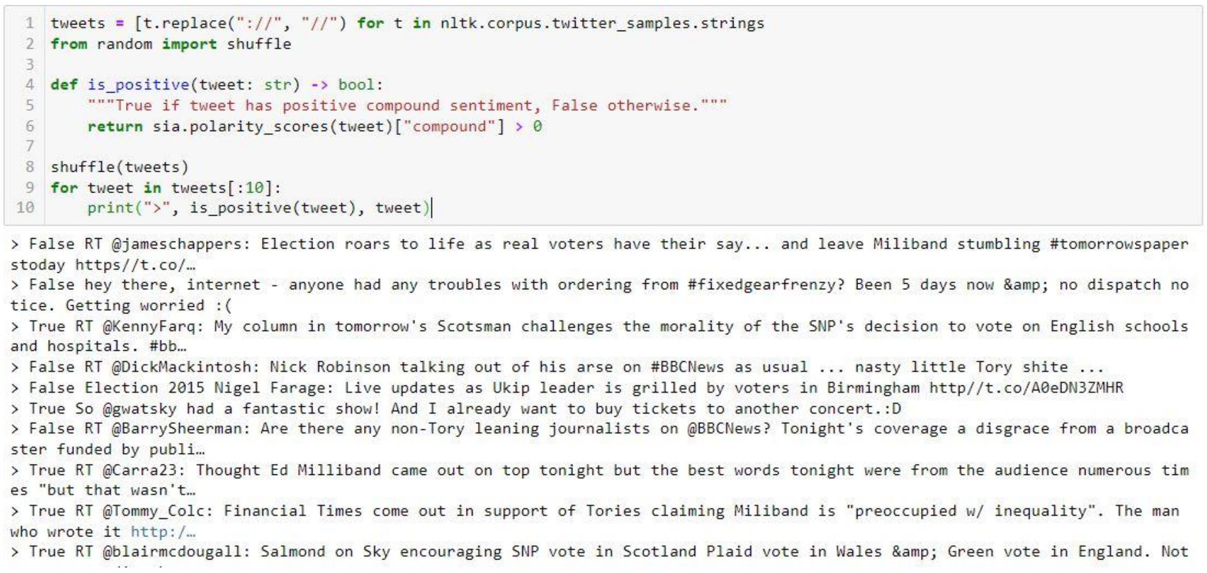
```
Accuracy is: 0.996
Most Informative Features
                      :) = True           Positi : Negati =   1011.9 : 1.0
                follower = True           Positi : Negati =     40.4 : 1.0
                followed = True           Negati : Positi =     22.9 : 1.0
                     sad = True           Negati : Positi =     19.4 : 1.0
                 welcome = True           Positi : Negati =     15.2 : 1.0
                  arrive = True           Positi : Negati =     14.9 : 1.0
              appreciate = True           Positi : Negati =     14.1 : 1.0
                   enjoy = True           Positi : Negati =     13.5 : 1.0
                     idk = True           Negati : Positi =     12.6 : 1.0
                   didnt = True           Negati : Positi =     11.9 : 1.0
None
```

Then we will use another resource vader_lexicon. We will get back a dictionary of different scores.



```
In [45]:  from nltk.sentiment import SentimentIntensityAnalyzer
          sia = SentimentIntensityAnalyzer()
          sia.polarity_scores("Wow, NLTK is really powerful!")

Out[45]: {'neg': 0.0, 'neu': 0.295, 'pos': 0.705, 'compound': 0.8012}
```

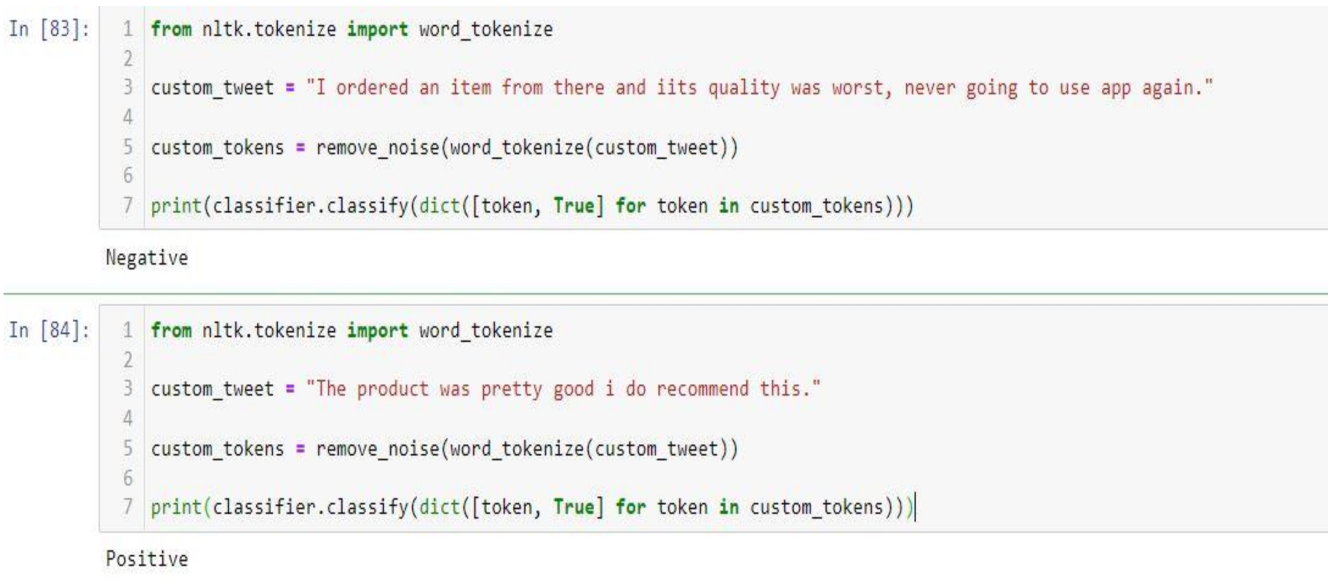
We will use polarity.scores() function to classify tweets.



```
tweets = [t.replace("://", "//") for t in nltk.corpus.twitter_samples.strings
from random import shuffle

def is_positive(tweet: str) -> bool:
    """True if tweet has positive compound sentiment, False otherwise."""
    return sia.polarity_scores(tweet)["compound"] > 0

shuffle(tweets)
for tweet in tweets[:10]:
    print(">", is_positive(tweet), tweet)
```

```
> False RT @jameschappers: Election roars to life as real voters have their say... and leave Miliband stumbling #tomorrowspaper
stoday https//t.co/…
> False hey there, internet - anyone had any troubles with ordering from #fixedgearfrenzy? Been 5 days now &amp; no dispatch no
tice. Getting worried :(
> True RT @KennyFarq: My column in tomorrow's Scotsman challenges the morality of the SNP's decision to vote on English schools
and hospitals. #bb…
> False RT @DickMackintosh: Nick Robinson talking out of his arse on #BBCNews as usual ... nasty little Tory shite ...
> False Election 2015 Nigel Farage: Live updates as Ukip leader is grilled by voters in Birmingham http//t.co/A0eDN3ZMHR
> True So @gwatsky had a fantastic show! And I already want to buy tickets to another concert.:D
> False RT @BarrySheerman: Are there any non-Tory leaning journalists on @BBCNews? Tonight's coverage a disgrace from a broadca
ster funded by publi…
> True RT @Carra23: Thought Ed Milliband came out on top tonight but the best words tonight were from the audience numerous tim
es "but that wasn't…
> True RT @Tommy_Colc: Financial Times come out in support of Tories claiming Miliband is "preoccupied w/ inequality". The man
who wrote it http:/…
> True RT @blairmcdougall: Salmond on Sky encouraging SNP vote in Scotland Plaid vote in Wales &amp; Green vote in England. Not
```

Now we will check how model performs on random tweets from twitter.



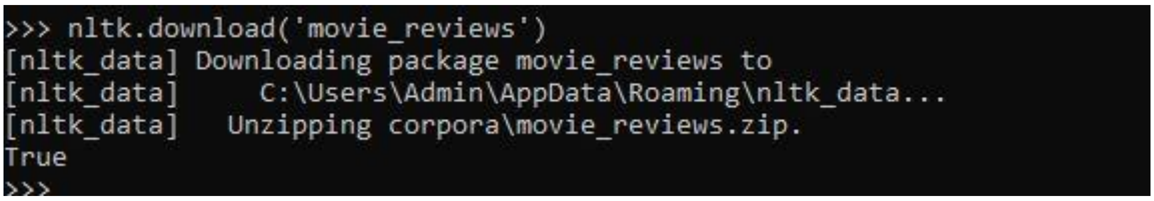
```
In [83]:
from nltk.tokenize import word_tokenize

custom_tweet = "I ordered an item from there and iits quality was worst, never going to use app again."

custom_tokens = remove_noise(word_tokenize(custom_tweet))

print(classifier.classify(dict([token, True] for token in custom_tokens)))
```

```
Negative
```

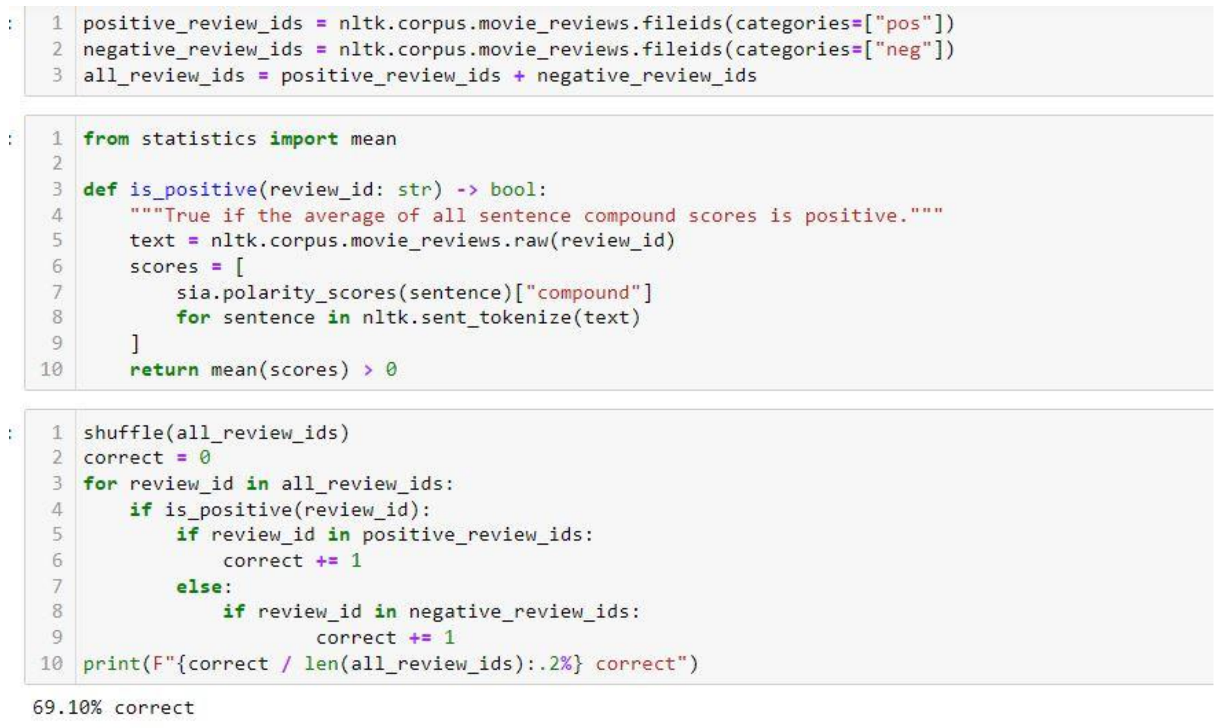
```
In [84]:
from nltk.tokenize import word_tokenize

custom_tweet = "The product was pretty good i do recommend this."

custom_tokens = remove_noise(word_tokenize(custom_tweet))

print(classifier.classify(dict([token, True] for token in custom_tokens)))
```

```
Positive
```

We will use another resource movie_reviews.



```
>>> nltk.download('movie_reviews')
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\movie_reviews.zip.
True
>>>
```



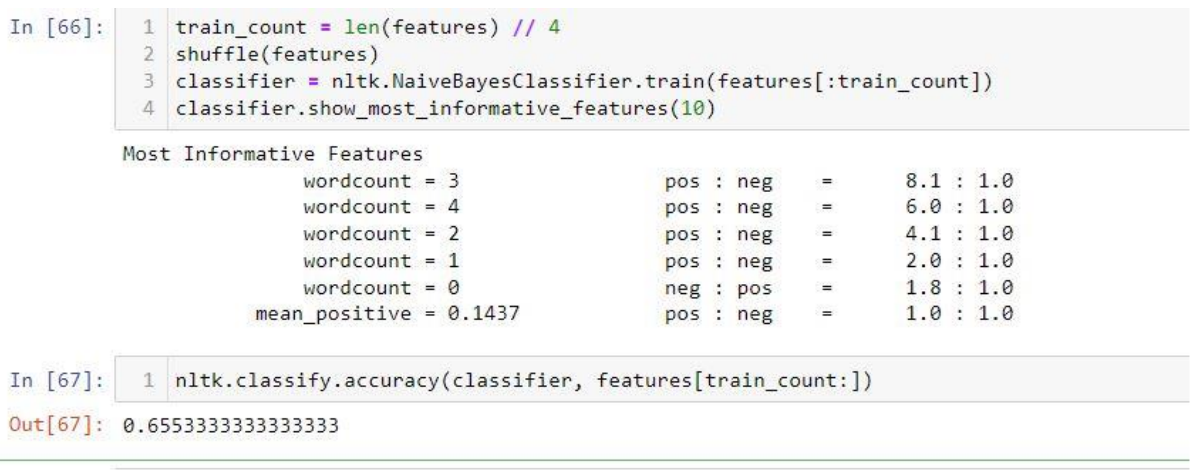
```
:   positive_review_ids = nltk.corpus.movie_reviews.fileids(categories=["pos"])
    negative_review_ids = nltk.corpus.movie_reviews.fileids(categories=["neg"])
    all_review_ids = positive_review_ids + negative_review_ids
```

```
:   from statistics import mean

    def is_positive(review_id: str) -> bool:
        """True if the average of all sentence compound scores is positive."""
        text = nltk.corpus.movie_reviews.raw(review_id)
        scores = [
            sia.polarity_scores(sentence)["compound"]
            for sentence in nltk.sent_tokenize(text)
        ]
        return mean(scores) > 0
```

```
:   shuffle(all_review_ids)
    correct = 0
    for review_id in all_review_ids:
        if is_positive(review_id):
            if review_id in positive_review_ids:
                correct += 1
            else:
                if review_id in negative_review_ids:
                    correct += 1
    print(F"{correct / len(all_review_ids):.2%} correct")
```

```
69.10% correct
```

We will be using train() method that involves splitting the feature set so that one portion can be used for training and other for evaluation.



```
In [66]:  train_count = len(features) // 4
           shuffle(features)
           classifier = nltk.NaiveBayesClassifier.train(features[:train_count])
           classifier.show_most_informative_features(10)
```

```
Most Informative Features
                   wordcount = 3                   pos : neg    =      8.1 : 1.0
                   wordcount = 4                   pos : neg    =      6.0 : 1.0
                   wordcount = 2                   pos : neg    =      4.1 : 1.0
                   wordcount = 1                   pos : neg    =      2.0 : 1.0
                   wordcount = 0                   neg : pos    =      1.8 : 1.0
               mean_positive = 0.1437              pos : neg    =      1.0 : 1.0
```

```
In [67]:  nltk.classify.accuracy(classifier, features[train_count:])
Out[67]:  0.6553333333333333
```