Sentimental Analysis

Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral.

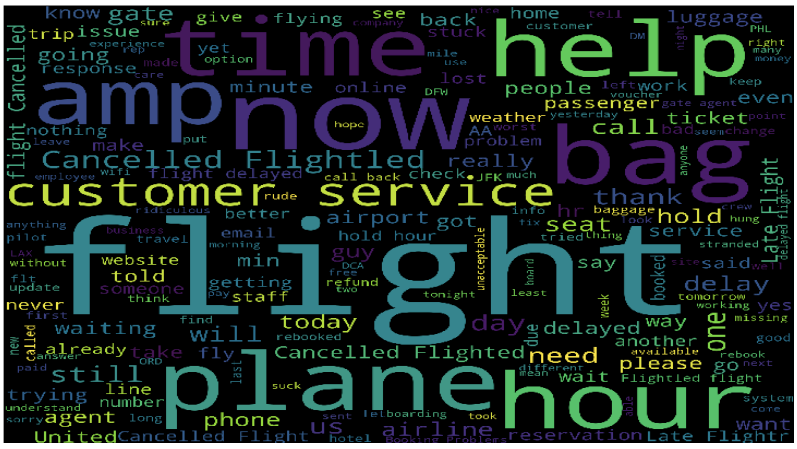
Steps involved in Analysis

1. First, we need to split the dataset in training and testing dataset to check our training accuracy on testing dataset. Testing dataset is 10% of original dataset.

2. I have dropped other columns except airline_sentiment because we need to analysis based on their positive and negative reviews.

3. Further I have dropped Neutral and Positive tweet because I need only negative shared experience.

4. Now they were ready for a WordCloud visualization which shows only the most emphatic words of the Negative tweets.



Algorithm:

I have used NLTK NaiveBayes classifier algorithm because It works fine on text classification problems. It relies on a very simple representation of the document (called the bag of words representation) and work perfectly with true positive and true negative cases.

They are probabilistic, which means that they calculate the probability of each tag for given text, and then output the tag with the highest one. The way they get these

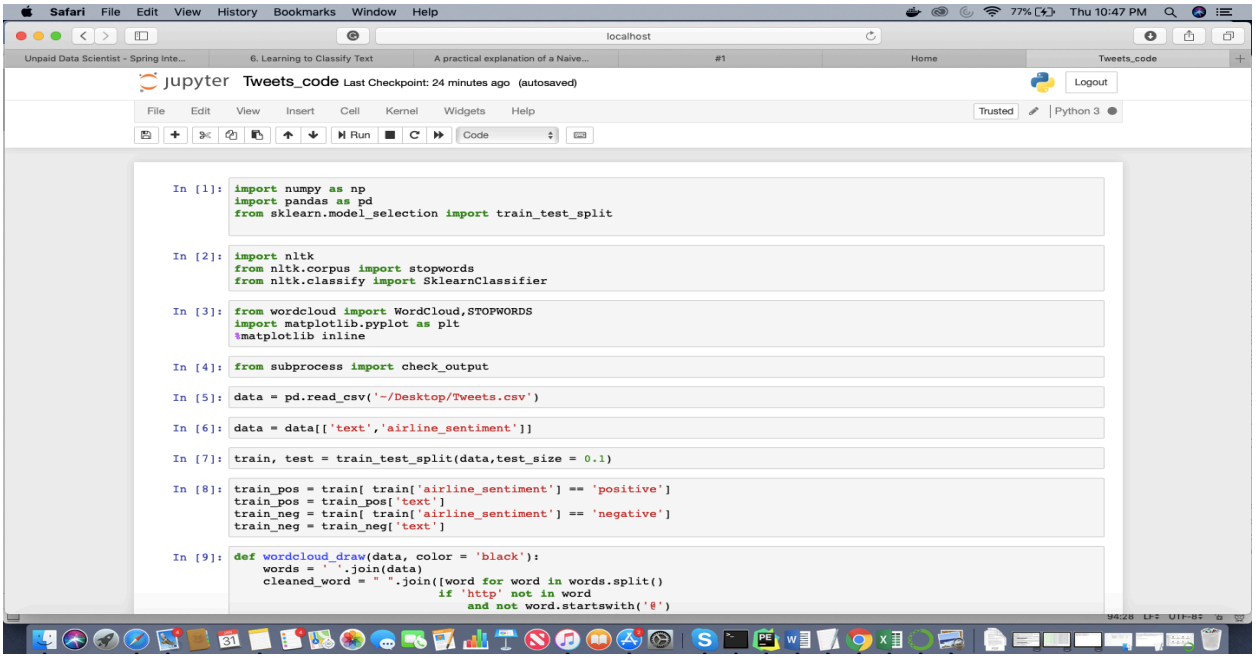
probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.

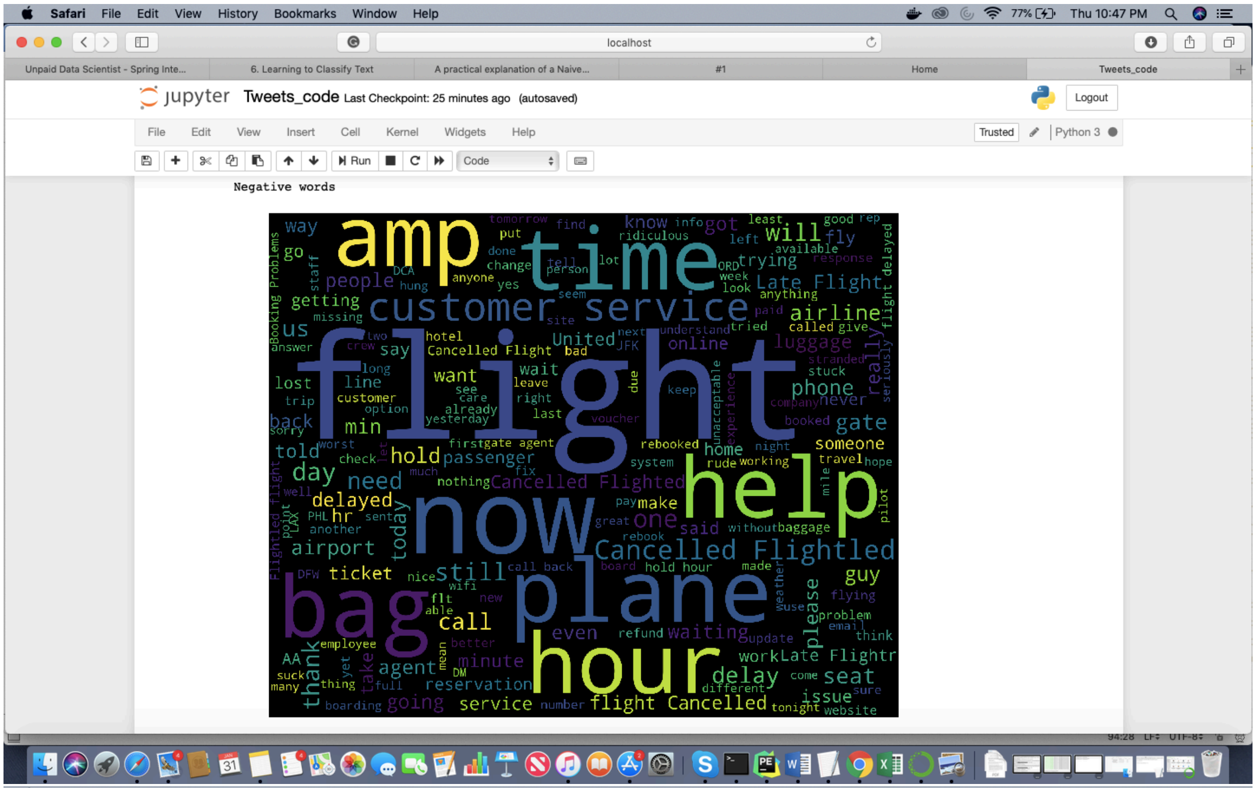
The analysis of the results (goals and conclusions)

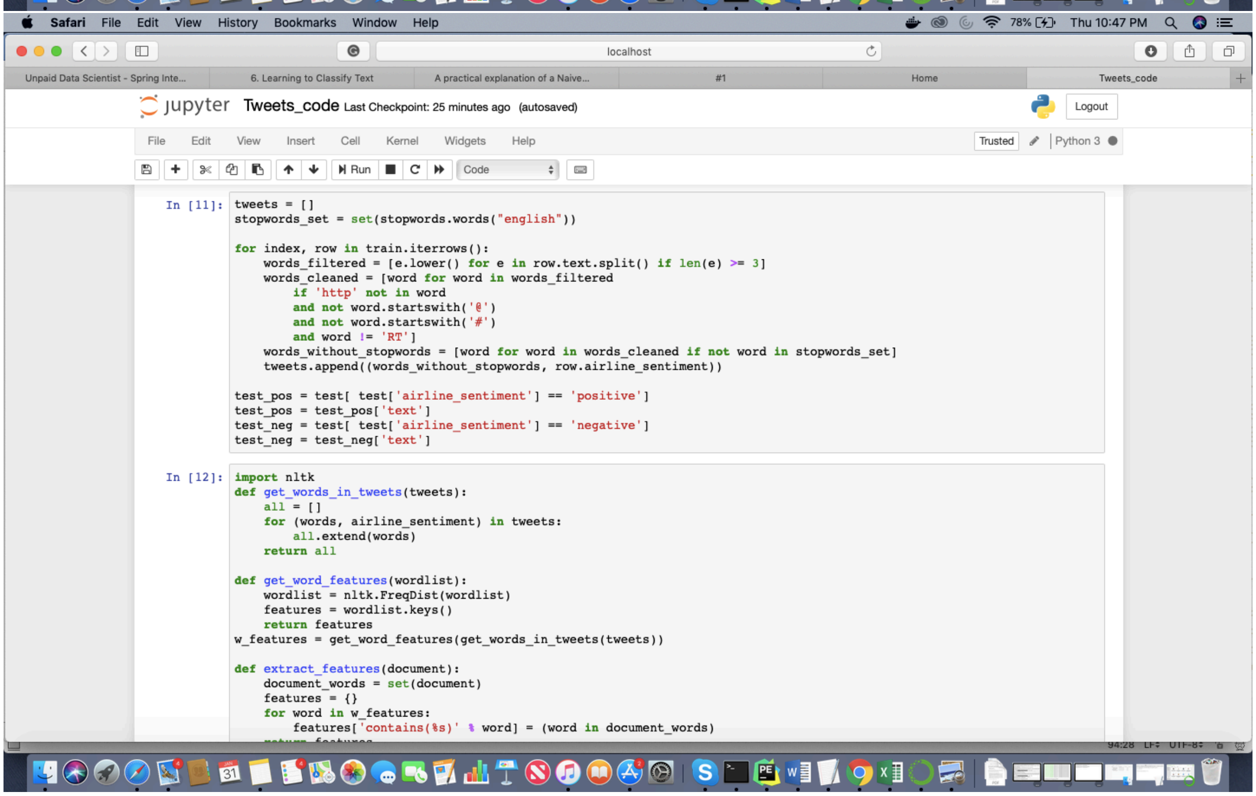
The classifier trained algorithm works perfect for negative comments on testing data.
Negative testing data's accuracy is much better than positive comments.
The problems arise when the tweets are ironic, sarcastic has reference or own difficult context.

Note: I used Anaconda-Navigator to work on python3.
Please find my working screenshot on jupyter notebook.

```
[Negative]: 842/795
[Positive]: 220/74
```








```
In [11]: tweets = []
stopwords_set = set(stopwords.words("english"))

for index, row in train.iterrows():
    words_filtered = [e.lower() for e in row.text.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in stopwords_set]
    tweets.append((words_without_stopwords, row.airline_sentiment))

test_pos = test[ test['airline_sentiment'] == 'positive']
test_pos = test_pos['text']
test_neg = test[ test['airline_sentiment'] == 'negative']
test_neg = test_neg['text']
```

```
In [12]: import nltk
def get_words_in_tweets(tweets):
    all = []
    for (words, airline_sentiment) in tweets:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_tweets(tweets))

def extract_features(document):
    document_words = set(document)
    features = {}
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
```

Thank You