For All of these algorithms the sci-kit learn library will be used

1) Naïve Bayes algorithms (included in Scikit)

- Based on Bayes Theorem

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

P(B|A): probability of B occurring if A is true
P(A|B): probability of A occurring if B is true.
P(A) and P(B) the probability of A and B respectively

- Pros:
  - These algorithms are simple to implement.
  - Easily trainable even with a small set of data
  - Fast
- The problem is that it assumes every word is independent
  - For example, for the words banana split, it will interpret the words "banana" and "split" separately instead of joining them and interpreting them as the dessert item.

2) Linear Support Vector Machines

- From the training data, SVM determines a line that separates the classifications. This line is called a hyperplane.
- If the new test data is on one side of the hyperplane then it is classified by the data on that side of the hyperplane and vice versa.
- Advantages:
  - Produces accurate fits
- Disadvantages:
  - It is a binary classifier (works when there are only 2 classifications)

Unsupervised Learning algorithms

K means clustering

- Groups data based on similarity and determines of a point is part of the cluster by measuring the distance of the pint to the centroid of each cluster.
- Uses Tfdif vectors
- Disadvantages:
  - More difficult to implement and algorithm is sensitive
- Advantages:
  - Doesn't need training data.

Maybe Try Mean Shift clustering but not main priority.

Before converting data from text to a form that is compatible with machine learning algorithms we must…

- Use stemming (or preferably lemmatization since it is more accurate but stemming is faster)
- Get rid of stop words
- Remove symbols such as hash tags and "@" signs
- Should we parse websites that are linked in a tweet?
- Found a library that cleans up twitter data https://github.com/s/preprocessor
-

Bag of words preprocessing algorithm in NLTK

- Counts the frequency of each word in a document
  - Easy to implement through nltk
    - Use count vectorizer
      - The count vectorizer output a tuple that includes the document that a word occurs in the position of that word and its frequency
        - E.g "This is just an example but no more examples"
          (1,5) 2 this says that the word example appears twice in document 1

Tf-idf representation

- Uses bag of words but also weights the word by how often in appears in the corpus because more rare words are more important. (may be the best preprocessing algorithm).
- In information retrieval, **tf–idf**, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.