Week 1:
Logistic regression, is a very important tool used in many applications in NLP. Logistic regression algorithms are particularly useful because they are easy to train and provide you with a good baseline result.

First we have to extract the features. Then we have to train your model. And then you have to classify the tweet based off your trained model.

- In order to do so, we first have to build a vocabulary and that will allow us to encode any text or any tweet as an array of numbers.
- With the sparse representation, a logistic regression model would have to learn n plus 1 parameters, where n would be equal to the size of your vocabulary and you can imagine that for large vocabulary sizes, this would be problematic.
- Keep track of the number of times, that's where it shows up as the positive class. Given another word we want to keep track of the number of times that word showed up in the negative class.
- The first feature would be a bias unit equal to 1. The second is the sum of the positive frequencies for every unique word on tweet m. The third is the sum of negative frequencies for every unique word on the tweet.
- Remove all the words that don't add significant meaning to the tweets, aka stop words and punctuation marks.
- Stemming in NLP is simply transforming any word to its base stem, which you could define as the set of characters that are used to construct the word and its derivatives.
- To reduce your vocabulary even further without losing valuable information, you'd have to lowercase every one of your words.

Implementation:
```
freqs=build_freqs(tweets,labels) #Build frequencies dictionary
X=np.zeroes((m,3)) #initialize matrix X
for i in range(m) #for every tweet
p_tweet=process_tweet(tweet[i]) #process tweets
X[i,:]=extract_features(p_tweet,freqs) #extract features
```

To train your logistic regression classifier, iterate until you find the set of parameters theta, that minimizes your cost function. Let us suppose that your loss only depends on the parameters theta1 and theta2.
We have to initialize your parameters vector theta. Then we use the logistic function to get values for each of your observations. After that, you'd be able to calculate the gradients of your cost function and update your parameters. Finally, you'd be able to compute your cost J and determine if more iterations are needed according to a stop-parameter or maximum number of iterations.