Question 1)

To start off this NLP project, the first goal is to get a sense of our data. To do this I needed to figure out the most common words to effectively get rid of any stop words that might not be helping us in the data. Looking at the most common words in the whole document, “the, and, a, I, is, to, of, was, this, it, in”, were in the top 10. These words we can get rid of as they won’t really add value to our modelling later. Now we have to build a stop words list, there is one in NLTK library which they caution not to use. Further investing why this might be the case, I looked at what is actually in the stopwords “English” library in the NLTK library. Turns out that some of these words might actually be useful to keep in the case of what we’re trying to solve. Words such as “not”, “nor”, “no”, “wouldn’t” which are in the stopwords list might be useful. So we can keep those in, and create our own stop words list that we will look for in the stop words list as well as the most common words and see what we can do without from there. Looking at some of the texts we can see there are some punctuation and dashes which can change a meaning of a word. If we are to use the count vectorizer that’s imported from NLTK we would lose these punctuation. Punctuations in between numbers such as “/” in “1/10” will mean a lot to us, so when we vectorize the words we can keep those in. In our case I chose to work with work pairs because if the document picks up the word “good” but doesn’t pick up the “not” in front of it, it might misclassify. Now that we know what we want to achieve with our bag of words we can go ahead and create a pipeline. I am handpicking some of the stop words in the ‘english’ stopwords library from NLTK but leaving some of the words that might be useful for our classification as mentioned above and more. We also checked the most common words and added those to the stop word lists that we think will not provide value to our classification. Words such as ‘characters’, ‘take’, ‘since’ and some others. We also pulled the words in bi grams with limited punctuations such as ‘-‘ and ‘/’ as described above. We then lemmatized our words which can help us reduce some redundancy in our words. Lemmatisation according to Wikipedia is defined as “is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.” I chose to use lemmatization instead of the porter stemmer because porter stemming sometimes getting rid of the end characters which can change up the meaning of a word. Below is some examples of the difference between porter, lancastor and lemmatiser:

Word	Porter	Lancaster	Lemmatiser
apples	appl	appl	apple
pears	pear	pear	pear
tasks	task	task	task
children	children	childr	child
earrings	ear	ear	earring
dictionary	dictionari	dict	dictionary
marriage	marriag	marry	marriage
connections	connect	connect	connection
universe	univers	univers	universe
universities	univers	univers	university

With n_gram being 1 we have 3620 features, when we change this to bi grams our features scale up to 12181, but when we change the range to 1,3 which I believe will help us during our modeling phase as we have more to compare. Regarding Tf-IDF I chose to keep both as doing some research online it ended being the most reliable.

Below are word clouds that show the most prevalent words, and prevalence is defined in this case by the frequency of the word. The figure on the left show the most common words in the positive reviews, and the figure on the right show the most common words in not positive reviews.

Figure 1 Word cloud of the most common words where the sentiment is positive

Figure 2 WordCloud of the most common words seen where sentiment is NOT positive

Question 2)

MODEL	TRAINING ACCURACY	LEADERBOARD ACCURACY	LEADERBOARD AUROC
BASELINE LOGREG (N_GRAMS = 1)	.94	0.833	0.912
BASELINE LOGREG (N_GRAMS = 1,2)	.972	0.828	0.91169
BASELINE LOGREG (N_GRAMS = 2,2)	.985	0.657	0.742
LOGREG (MAX_ITER = 10, BIGRAM)	.943	0.83	0.912

Now that we have our TF-IDF set we can start doing logistic regression. We start with a baseline model and we can check different n_grams and how that performs in the leaderboard. Looks like the unigram did the best in terms of leaderboard error rate but the combination of bigrams and unigrams was a close second. Just the bigrams itself wasn't really that effective and looks like it overfit the training data, so from now when we use parameter tuning we will use on the combination of unigrams and bigrams as this was not only close to the n_grams of 1 accuracy but it also had a great training accuracy. Let's start with the iterations parameter. Looks like iterations in this case doesn't help must as after the 10th iteration the accuracy stays the same.