There are words that take place in several documents from both classes (negative and positive), so they do not give relevant information. To overcome this problem, I used a technique *called term frequency-inverse document frequency* (td-idf). The technique uses not only the frequency of a word, but also the uniqueness of the word, therefore giving a more accurate depiction of the classification. To increase the accuracy of the model in prediction, I tried adding bigrams, where tokens represents pairs of consecutive words, hence allowing better classification prediction from the model.

In choosing which solver to use, I understood that only saga, sag, lbfgs and newton-cg handle multinomial problems and hence they defined my scope of choice. From what I learnt, saga and sag work faster with large data-sets and taking into consideration our dataset size, I assumed sag would work well. The fact that only but the saga solver can support the cross-entropy loss function also informed the choice of sag as a solver.

Pandas was used to process the data, this is because it provides a comprehensive inherent processing capacity that I personally would take a lot of time to achieve. The SKLearn library was comprehensively used because it allows the user to access plenty of features e.g. tf-idf features. It is this aspect of the SKLearn library, including but not limited to its inherent classes and vital processing tools that made it the easy choice.

Logistic Regression Model with unnormalized sentences

Accuracy = 0.8837718926433935

Logistic Rregression with normalized sentences

Accuracy = 0.9089004258495784

Naive Bayes with unnormalized sentences

Accuracy = 0.8859695396919935

Naive Bayes with normalized sentences

Accuracy = 0.8948367952522257

The results for each classifier seemed to come out as expected, with the normalized classifiers of each type of classifier (Naïve Bayes (nn) and Linear Regression (nl)) producing a higher percentage of accuracy as compared to their unnormalized counterparts (Naïve Bayes (un) and Linear Regression (ul)). This proves that normalization of data increases the accuracy of the classifier; from this it can also be postulated that if I had done more normalization, even used other n-gram techniques, the accuracy of the classifier would have been increased. Personally, I should continue working on the classifiers in order to generate a better method of accuracy and evaluation which takes into consideration more than just accuracy but also other measures (an example would be to use/generate a confusion matrix for proper result/accuracy analysis).

I also noticed that, by tweaking test conditions and adding some other features to the data, the data can be improved and the output(accuracy) made better.

I also tried to tweak a few parts, even thought the code seemed to be working well. I feel the way the model is implemented determines its accuracy, this model was implemented using a multi-class, and to test this I will also work on separate models for each using either singular classes or functions.