Designed naive Bayes classifier for sentiment analysis:

Computed prior probability of positive and negative class and the probability likelihood of each word (in the vocabulary of training data) being in positive and negative class

1. Computed prior probabilities of positive and negative class **[P(+), P(-)]**
2. Each sentence in the training data sets is tokenized into words are converted to lower case (word in lower case and no punctuations at the end of a word)
3. Created two separate dictionaries to store positive and negative word counts. Generated a vocabulary list from the positive and negative dictionaries
4. Designed a Numpy matrix of size (vocabulary, 2), where each row corresponds to a word and first column holds the probability of word being in '-' class and 2nd column for holding the probability of word being in '+' class. To avoid the zero probability likelihood term, matrix is initialized with ones when created (Laplace smoothing) and each entry stores the word_count + 1 and the probability is computed.
5. Highly frequent words are considered as stop words.

Test data:
For each sentence in the test data, computed the probability of falling under '+' and '-' class

Each sentence:
1. Tokenized and word is converted to lower case and removing the punctuations.
2. Probability of sentence is computed by multiplying probability of each word and prior probability.
3. If a new word occurs in the test data, we just ignore it.
4. POS or NEG class is assigned based on whichever is highest for that sentence

Accuracy:
In the training phase, I split the training set into 60% training and 40% development data and the accuracy ~ 90%

Libraries used Numpy, pandas, collections