In completing this assignment, the implementation below is geared towards the goals:

1. Implement naïve bayes from scratch
2. Use the given data (movie reviews) to train and develop the model, and apply the trained model on the test dataset split from the given data
3. Evaluate and attempt to up the accuracy of the algorithm with the adjustable parameters

Data processing

1. The positive and negative review entries are read and stored into two lists.
2. The data is randomly split into 70% train set, 30% dev set, and 30% test set as specified. This results in 6 lists – pos\_trainSet, pos\_devSet, pos\_testSet, neg\_trainSet, neg\_devSet, and neg\_testSet.
3. The lists are cleaned – stopwords, symbols and punctuations are removed, and letters are lowercased. This results in the 6 lists only containing strings of meaningful words.
4. The strings of words, each representing a movie review entry, are vectorized using the bag of word tool CountVectorizer from scikit-learn. To utilize the most frequently used words, the max\_feature of the CountVectorizer is initialized to 100 (which will later be adjusted for dev dataset).
5. The resulting data from the above process is pos\_train\_vect, pos\_dev\_vect, pos\_test\_vect, neg\_train\_vect, neg\_dev\_vect, and neg\_train\_vect, all of which are arrays of 100-dimension vectors representing the count of the most frequently used words from each dataset. While their labels are all known (positive or negative), only pos\_train\_vect and neg\_train\_vect will be applied along with the labels for training, and pos\_dev\_vect and neg\_dev\_vect’s label for tuning, and pos\_test\_vect and neg\_test\_vect’s label for comparison with the prediction.

Following the above pre-processing of the data, the algorithm described below is applied. The description does not necessarily include all the steps applied in the script as some methods are wrappers for formatting purposes. See script for annotations.

Algorithm

1. Calculate the means and standard deviation of all the attributes pos\_train\_vect and neg\_train\_vect.
2. The probability of one entry belonging to a classification is calculated based on one attribute.
3. The combined probability is calculated from all attributes with multiplication, and the same process is done for all entries.
4. With all attributes along with the expected label of the datasets (pos\_train\_vect and neg\_train\_vect), the probabilities of each entry belonging to each classification from the input vector is calculated to make a prediction. The input vectors in question are pos\_dev\_vect and neg\_dev\_vect.
5. Accuracy of each predictions are obtained by calculating the percentage of the right outcomes.

As a result, the accuracy of the predicted label is listed below for both positive and negative reviews. As the algorithm itself doesn’t appear to have parameters to tweak the result, the only visible parameter to tune is the number of max\_feature from the CountVectorizer. Thus three numbers of max\_feature is used to compare the accuracy:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table 1: Accuracy of Naïve bayes on movie review labels with varying max\_feature (from CountVectorizer) on Dev dataset (rounded up to integer)  (output available in screenshot) | | | | | | |
|  | Max\_feature | | | | | |
|  | **50** | | **100** | | **500** | |
| Positive | 33 | Average: 44 | 41 | Average: 47 | 51 | Average: 52 |
|  | 50 | 49 | 51 |
|  | 49 | 48 | 47 |
|  | 38 | 47 | 59 |
|  | 48 | 52 | 52 |
| Negative | 53 | Average: 60 | 47 | Average: 56 | 56 | Average: 51 |
|  | 62 | 57 | 44 |
|  | 68 | 54 | 55 |
|  | 60 | 61 | 60 |
|  | 57 | 61 | 40 |
|  |  | Total: 104 |  | Total: 103 |  | Total: 103 |

The last row on table 1, the total accuracy score, is an attempt to obtain the optimal number of max\_feature in order to choose one to apply to the test dataset, as the resulted accuracy from the positive vs negative reviews are quite different. Even though the max\_feature at 50 only wins by a margin of 1, this will be chosen to apply the algorithm on the test dataset which results in the table below:

|  |  |  |
| --- | --- | --- |
| Table 2: Accuracy of Naïve bayes on movie review labels with max\_feature = 50 on test dataset (rouded up to integer)  (output available in screenshot) | | |
| Positive | 61 | Average: 61 |
|  | 71 |
|  | 67 |
|  | 43 |
|  | 64 |
| Negative | 64 | Average: 60 |
|  | 66 |
|  | 64 |
|  | 40 |
|  | 64 |
|  |  | Total: 121 |

As shown in table 2, the resulted accuracy on the test datasets have improved compared to the prior results on the development sets, and the gap from the results of positive vs negative is closing.

Conclusion:

1. The overall accuracy of predicting the label for negative reviews is higher. In the cases where the result from predicting positive reviews is higher, it is only so by a small margin (as in table 2 and last column in table 1). While it is premature to say a trend is forming, a closer look at the data would be the next step. Specifically, it is possible that certain words are repetitively used in negative reviews, if such a trend exists.
2. From the above implemented algorithm, there is no tweakable parameter to potentially up the accuracy except for the given data. For that purpose, it is also worth looking into other implementations of naïve bayes to get more comparable results on the same datasets.