NAIVE BAYES CLASSIFIER

Results after Text Preprocessing -

Validation Set-

Final Accuracy Validation = 0.802 +/- 0.034
Final Validation F-Score = 0.812 +/- 0.039

Test Set -

Accuracy on test set = 0.82
Precision = 0.7857142857142857
Recall = 0.88
Fscore = 0.830188679245283

Evaluation Results on Validation Set across the 5 folds of cross-validation -

Accuracy on validation set [0.81875, 0.8125, 0.84375, 0.74375, 0.7924528301886793]

Precision [0.7857142857142857, 0.797752808988764, 0.8295454545454546, 0.651685393258427, 0.7857142857142857]

Recall [0.8571428571428571, 0.8554216867469879, 0.8795180722891566, 0.8529411764705882, 0.8651685393258427]

Fscore = [0.8198757763975155, 0.8255813953488371, 0.8538011695906433, 0.738853503184713, 0.8235294117647058]

Results before Text-Preprocessing -

Validation Set

Final Accuracy Validation = 0.737 +/- 0.022
Final Validation F-Score = 0.757 +/- 0.025

Test Set

Accuracy on test set = 0.78
Precision = 0.7545454545454545
Recall = 0.83
Fscore = 0.7904761904761904

Evaluation Results on Validation Set across the 5 folds of cross-validation -

Accuracy on validation set [0.75, 0.71875, 0.775, 0.71875, 0.7232704402515723]

Precision [0.6907216494845361, 0.7065217391304348, 0.7640449438202247, 0.6263736263736264, 0.7272727272727273]

Recall [0.8701298701298701, 0.7831325301204819, 0.8192771084337349, 0.8382352941176471, 0.8089887640449438]

Fscore = [0.7701149425287357, 0.7428571428571429, 0.7906976744186045, 0.7169811320754716, 0.7659574468085106]

Observations -

From the above results, it is evident that the Naive Bayes Model's Accuracy and F-Score improved significantly after Text Preprocessing.

Improvement in Accuracy and F-Score after Text Preprocessing is approximately around 4%.

Steps taken in Text Preprocessing -

- All the words in the vocabulary as well as in the test sentences were converted to lowercase.
- All the numbers or digits in the vocabulary as well as in the test sentences were removed.
- All the punctuations or symbols in the vocabulary as well as in the test sentences were removed.

Algorithm Flow and Design Decisions -

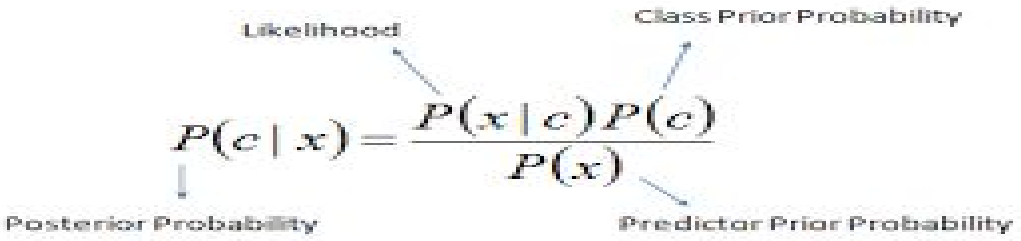
We took the dataset containing the text and the corresponding sentiments of those texts. First, we splitted the data into two lists with the first list having sentences and the second list containing the sentiments (0 or 1)for those sentences, where 0 represents that a particular sentence has a negative sentiment and 1 represents that a particular sentence has a positive sentiment.

Now, we **splitted the data (lists) in a 80:20 ratio**, where training data contains 80 percent of the original sentences and testing contains 20 percent of the sentences.

Now, **a 5-fold cross validation technique** was applied, dividing the train set further into 5 sets, out of which 4 sets were used as the training set and 1 set was used as the validation set. This division was done 5 times, with assigning a different set as the testing set everytime.

For building the vocabulary, a dictionary was made containing each unique word as key and a list containing counts of positive and negative sentiment respectively denoting the number of times that word appeared in the vocabulary as a positive sentiment word and a negative sentiment word.
Laplace Smoothing was also performed by adding a constant factor of 1 to the positive as well as negative count of every word. This was done to eliminate the possibility of a zero probability (as minimum count is 1 for every word).

For the testing and validation part, the same preprocessing technique is applied on the test sentence and 2 probabilities are calculated both for negative and positive sentiment class. The class with maximum probability is assigned as the sentiment for that sentence. The probability calculation is done using the concept of Bayes Theorem.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Here, c-> class(positive or negative), P(c) is the prior probability which is calculated as (if c1=1, P(c)=count of pos. sentiments/ count of total sentiments), xi's represent the words of the sentence (taken as independent of each other.)

Probability of every word for both classes is calculated from the preprocessed dictionary and is multiplied according to the above formula. If a word is not present in the dictionary , then its probability is assumed to be 0.5(for both classes).

Conclusions -

- Naive Bayes assumes independent predictors and if that assumption is correct, a Naive Bayes classifier will perform better as compared to other models.

- It also requires a small amount of dataset for training. The training period is also less.

- It was theoretically and practically easy to implement, and takes negligible training time.

- Laplace Smoothing by adding a constant factor of 1 to all the positive and negative sentiment count for all sentences eliminated the possibility of zero probability. So, now the prediction being made is more accurate and reliable.

- Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it is almost impossible that we get a set of predictors which are completely independent.

- If a word is in the test data set, which was not present in the vocabulary, then the model will assign a 0.5 probability to both the classes.