Report Naive Bayes from Scratch

Implementation

1. **Converting the text file**: The dataset text file is converted to a pandas dataframe. Each sentence in the dataset goes through some preprocessing steps. The preprocessing steps include converting to lowercase, removing punctuations, numbers and stop words.
2. **7-fold Cross Validation**: Next, the data frame is divided into 7 folds. For each of the 7 iterations, one of the sets is considered as the testing set and the rest of the data is used in training.
3. **Count Vectorization**: An array of unique words is derived from the training data. Each sentence in the training set is now fitted onto this array of unique words, giving a matrix that will be used for training.
4. **Fitting and evaluating the model**: Once we have a matrix representation for our training and testing data, we can fit the data into our model and use the Bayes theorem formula to derive the probabilities that a certain product review in our testing set is of positive sentiment. The predictions are compared to the actual labels for the testing data and accuracy is reported.
5. **Deriving the final accuracy**: Once we have the accuracies for each of the 7 folds (testing sets), we can take their mean and report it. This will be the accuracy of our model.

Working

We loop through the 7 different folds, create and fit the training data for that fold and evaluate the accuracy for each fold. Finally, we take the mean of all the accuracies and report it as the final accuracy of the model.

One major assumption of the model (and using count vectorization) is that words occur in product reviews conditionally independent of each other.

We use the following formula obtained from Bayes Theorem to predict whether the sentence is of positive sentiment:

$$P(+ \mid Sentence) = \frac{P(Sentence \mid +)\ P(+)}{P(Sentence)}$$

where:

$$P(Sentence) = P(Sentence \mid +)P(+) + P(Sentence \mid -)P(-)$$

Now under our assumption that the occurence of words in a sentence is conditionally independent, we can say that:

$$P(Sentence \mid +) = \prod_{word\ \in\ Sentence} P(word \mid +)$$

and

$$P(Sentence \mid -) = \prod_{word\ \in\ Sentence} P(word \mid -)$$

To find the probability that a word appears given the class, we can use the **Laplace Smoothing formula**:

$$P(word \mid +) = \frac{count(word, +) + k}{count(+) + k|V|}$$

where:

$count(word, +)$ is the number of times that word occurs in the positive training sentences,

k is a Laplace Smoothing constant,

$count(+)$ is the total number of words occurring in the positive class,

$|V|$ is the number of words occurring in our vocabulary.

Accuracy

Accuracy for the 7 folds is given below.

Accuracy for fold 1: **83.80%**
Accuracy for fold 2: **89.51%**
Accuracy for fold 3: **79.72%**
Accuracy for fold 4: **81.81%**
Accuracy for fold 5: **78.32%**
Accuracy for fold 6: **81.11%**

Accuracy for fold 7: **75.52%**

Mean Accuracy: **81.40%**

Limitations of Naive Bayes Classifier

- We have made the assumption that the occurence of words is conditionally independent of each other. However, in real life, this isn't always true as the occurrence of some words may depend on the occurrence of other words.
- If the model encounters a word in the testing set that wasn't a part of the training data, it will not be able to make a proper prediction. This is known as the **Zero Frequency Problem**.
- Naive Bayes requires a relatively larger training dataset to make accurate predictions.
- Using generative models like Naive Bayes is usually more complex than using a discriminative model as we have to learn how the data is formed and what makes the data part of a particular class instead of just learning what differentiates the classes.