**BITS F464**

**Machine Learning - Assignment 1**

**NAIVE BAYES**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Praneetha Vaddamanu** | **2017AAPS0319H** |

**Problem Statement and Methodology**

The given task is to implement the Naive Bayes algorithm for classifying a given set of reviews into positive and negative classes. It is a binary classification task, and we apply a probabilistic discriminative method. By considering each review as a set of features for one data instance, there are two ways to approach the problem. We can take the features as words, with their respective count values in a class (**bag of words method**), or apply the **binary method** approach taking a feature as the existence of a word (0 or 1) in a class. We apply both methods and compare accuracies. We also mention all preprocessing steps applied on the original dataset, as well.

**Dataset Description and Preprocessing**

It is required to test the implementation on the dataset provided in datasets/a1\_d3.txt.

The dataset consists of 1000 examples, and each example is a review provided in text format, followed by the corresponding label classifying it as a positive (1) or negative (0) review.

The preprocessing steps we follow are:

1. Converting each review to lower case
2. Removing all punctuation in the review, such as ‘, “, ., !, ?, etc
3. Removing symbols such as &,\*,^,%, etc

It is observed that these steps have improved the performance of the algorithm significantly, as it brings all words in the review to a common representation, allowing the identification of recurring words to be more accurate.

**Metrics for Performance Evaluation**

**Accuracy**

The accuracy calculated as,

captures how many data points (as a fraction of the total number of data points fed to the algorithm) were classified correctly. It ranges between 0 and 100 %. A higher accuracy is indicative of better performance by the model.

Relying only on accuracy to judge model performance however, may, in some cases, be misleading. Hence, we also calculate the F-score in order to better evaluate the model’s performance.

**F1-score**

The F1-score is calculated as,

where

,

and

.

Recall captures how many data points were correctly classified as positive points, as a fraction of the total number of actual positive data points in the dataset.

Precision captures the number of data points correctly classified as positive points, as a fraction of the total number of points classified as positive by the model.

Ideally, we would want a high recall, as well as high precision, but often there is a tradeoff between the two. In order to capture correctly the effect of both recall and precision on the evaluation of performance of the model, we use the *F1-score* which is the harmonic mean of recall and precision.

The F1-score ranges between 0 and 1. A higher F1-score is indicative of better performance by the model.

**Bag of words model**

The probability of a word belonging to a particular class *P(c|w)* may be modelled as the posterior probability obtained by multiplying the prior probability of that class *P(c)* with the likelihood of the word occurring in that class *P(w|c)*. The probability of the entire document belonging to that class is then estimated as the independent product of the likelihoods of each of the words, multiplied by the prior probability (naive assumption).

In the bag of words model, each document is modelled as a set of words, with each word having a count value, indicating the number of occurrences in the document.

The likelihood is then estimated as the ratio of the count of that word in the class, to the total count of all words in the class.

The prior class probability is estimated as the ratio of the number of documents (training examples) belonging to the class, to the total number of documents in the training set.

We also use **Laplace Smoothing** to increase accuracy. This adds a term of 1 to the numerator of the likelihood of each word belonging to the class, and a term of the order of the vocabulary length to the denominator. Thus, even if a word in the test sentence exists in the vocabulary, but not in the positive class, it will not force the likelihood of the document belonging to the positive class to zero. This would have happened had we not used Laplace Smoothing.

In order to prevent numerical overflow, we use the **log likelihood and log prior,** and add the log likelihoods of all the words in the document to the log prior of the class, for deciding the class of the document.

Words in the test sentence but not present in the vocabulary are ignored, and do not affect the prediction.

Whichever class results in a higher value of the abovementioned computation, is the class assigned to the document as its predicted class.

Using these techniques, the results obtained are as follows:

By **5-fold cross validation :**

These results are obtained by fixing the seed value for numpy.random.

**Training Accuracy = 97 %**

**Training F1-score = 0.9714**

**Test Accuracy = 81.1 土 1.5297 %**

**Test F1-score = 0.8157 土 0.01639552**

**Binary feature model**

The methodology for calculation of likelihood of a document belonging to a class and prior of that class is the same as in the bag of words model, the only difference is that the likelihood of a word is no longer calculated by taking its frequency in the class. Here, words can take values of only 0 or 1 (binary), indicating their presence or absence in a document. Thus, each document becomes a binary vector of dimension equal to the length of the vocabulary.

The likelihood of a word in a class is now calculated as the ratio of the number of documents that the word is present in, in that class, to the total number of documents in the class.

Here also, we use **Laplace Smoothing,** and add 1 to the numerator and 2 to the denominator of the likelihood ratio for a word in a class. This ensures that the absence of a word from a particular class does not force the entire likelihood of the document in that class to zero.

We also use the **log likelihood and log prior** in calculation to avoid numerical overflow.

Words in the test sentence but not present in the vocabulary are ignored, and do not affect the prediction.

Using these techniques, the results obtained are as follows:

By **5-fold cross validation :**

These results are obtained by fixing the seed value for numpy.random to be the same as that for the bag of words model.

**Training Accuracy = 96.875 %**

**Training F1-score = 0.96863237**

**Test Accuracy = 81.8 土 1.02956301 %**

**Test F1-score = 0.8166 土 0.01639552**

**Comparison**

As seen from the results, the **binary feature model** performed better than the **bag of words model** by

0.7 %, resulting in a maximum test accuracy of 81.8 %, and test f1-score of 0.01639552.