BIG DATA (45-980) Project

Political Blogs Classification using Naive Bayes Approach

Submitted By:
Naman Kumar (namank)
Fahad Islam (fi)

Table of Contents

1.0 Executive Summary

In today's world, data classification based on different features has become a very vital task, specifically when you are dealing with "Big Data". Data Classification is everywhere on the internet like identifying spams and separating them, classifying similar news articles, etc. As the amount of data increases, it becomes very important to automate this data classification process. Different machine learning algorithms have been implemented to do this task automatically.

In this project, we have worked on classifying political blogs into two types: Conservative blogs and Liberal blogs. We were able to get 120 blogs in the form of text files with their correct label, that is whether they are conservative blogs or liberal blogs. Then, we split the dataset into two sets: Training set, to train our classifier and Testing set, to test our classifier. Then we compare the predicted labels on the testing set to their actual labels to find the accuracy of our classifier. Also, we repeat this number of times by randomly splitting the dataset into training set and testing set and then finding the accuracy of the classifier for each case. There are different classifiers which can be using for this type of binary classification like Logistic Regression, Naive Bayes Approach, Neural Networks or Support Vector Machines. For the purpose of this task, we have decided to go with Naive Bayes Classifier[1] because it is not a very complex algorithm and it gives us pretty good accuracy. Once, we have the baseline algorithm ready, we do few extensions to improve the accuracy of our classifier like removing useless words and performing smoothing. Finally, using the Naive Bayes Classifier, we were able to classify the blogs into Liberal and Conservative blogs with more than 80% accuracy.

There can be many advantages of this algorithm. Firstly, as the number of blogs increases it becomes very difficult to go through each blog and classify them manually. Secondly, using a machine learning algorithm for classification will really speed up the complete process of blog classification. Thirdly, this can be easily extended for classifying blogs into more than two types.

2.0 Introduction

"Big Data" is a term used for a collection of data sets that is so large that the traditional data handling tools and techniques cannot manage and process it. Specialized software and techniques are used to handle Big Data. The need also arises when the data gets piled up to exabytes rapidly in real-time. The field of Machine Learning has proven to be very useful in solving many Big Data problems and providing answers to several Big Data queries. One of its applications is in the field called Sentiment Analysis. Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Our work focuses on exploiting these techniques to extract useful information out of political blogs.

In the present world the information from the political blogs and political blog directories are smartly used for research and analysis in political campaigns. We observe that blogs are hard to navigate through to find relevance and manipulating huge sets of unordered blogs is cumbersome. Different web blog directories provide categorized blog listings but most of them are manual and inefficient. When these databases grow rapidly it gets hard to manage and organize such data bases. Using the techniques of Machine Learning and Natural Language Processing the categorization process of these blogs can be automated.

For this project we will classify political blogs. Political blogs can be classified into different categories such as liberal, conservative, libertarian, issue specific, region specific etc. For the scope of our project we will do binary classification for classifying a blog as liberal or conservative.

3.0 Project Implementation

Here, we have obtained 120 political blogs in the form of simple text files. Out of 120 total political blogs,

1. Conservative blogs = 64
2. Liberal blogs = 56

We already know the labels of all the blogs. Our first task is to separate the blogs into two sets:

1. Training set: Training set basically consists of set of blogs and their corresponding outputs or labels. We use this set along with a Machine Learning algorithm to train our classifier which can be used to predict the label of new blogs.
2. Testing set: Testing set consists of set of blogs which can be used to test the accuracy of the classifier. We run our classifier on the blogs in the testing set and compare the predicted label with the actual label to see how accurate is our classifier.

The second task is to select a good Machine Learning algorithm. There are different algorithms which can be used:

1. Linear/ Logistic Regression
2. Bayes Classifiers
3. Neural Networks
4. Support Vector Machines

We decided to use Bayes Classifier for this task because it is not as complex as Neural Networks or Support Vector Machines and it gives us pretty good accuracy also.

3.1 Naive Bayes Classifier

Naive Bayes Classifier is a probabilistic classifier based on Bayes Theorem, see Figure 1. It assumes that value of a particular feature is independent of other features given the class variable. This is a reasonable assumption in our case. One of the extensions of Naive Bayes can be Full Bayes Classifier which treats the features as dependent on each other.

In Naive Bayes Classifier, we have to calculate the posterior probabilities for both conservative blogs and liberal blogs as shown in Figure 1.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

Figure 1. The formula to calculate the posterior probability

Once, we get the posterior probabilities for both types of blogs, we compare which one is more and assign that corresponding label to the blog.

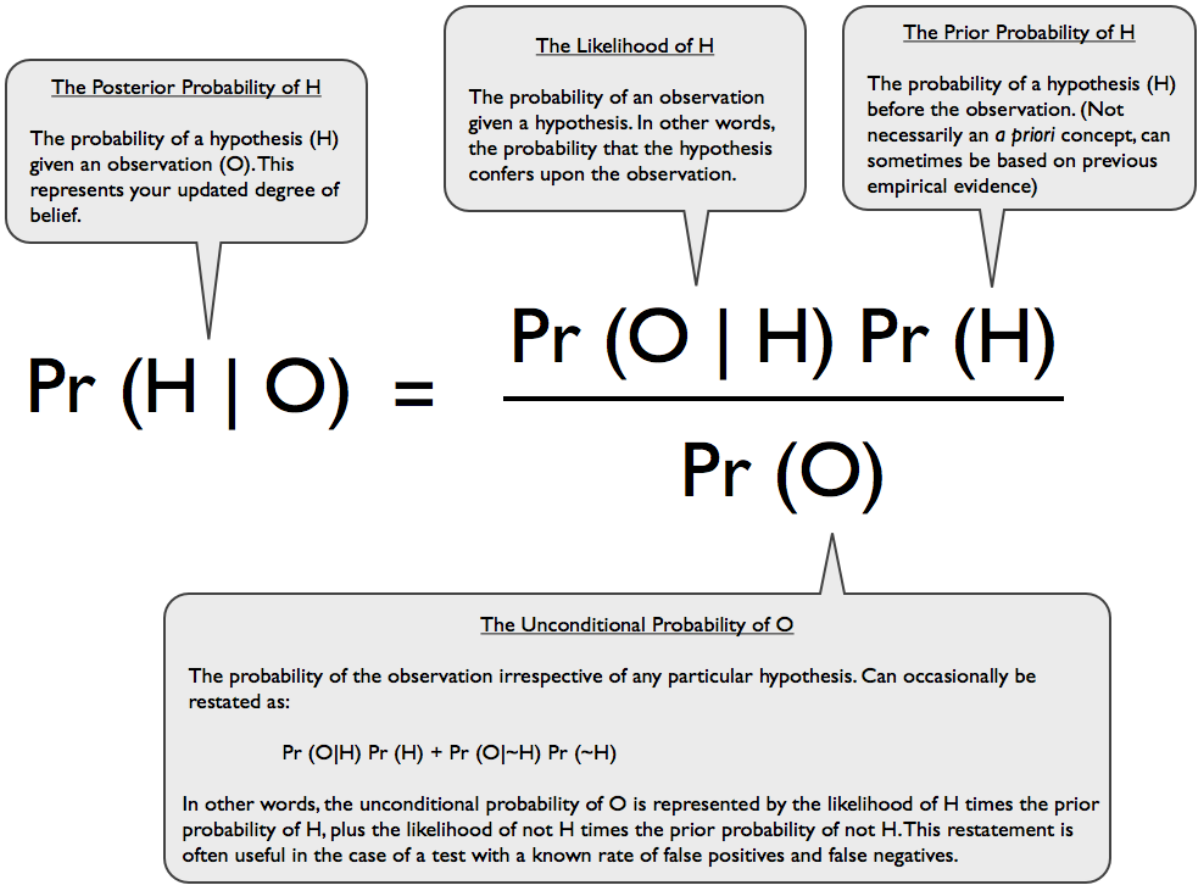

Figure 2. Bayes Theorem

3.2 Training

Once, we have decided which machine learning algorithm to use, the next step is to train the classifier on the testing set. The following steps are done:

1. Collect all words and other tokens that occur in $Examples$.

   $Vocabulary$ = Set of all distinct words and other tokens occurring in any text document generated from blogs.

2. Calculate the probability of each label ($P(v_j)$ and the probability of each word in the vocabulary with a given label $(Pw_k|v_j)$.

a) For each label v_j,

i) $Blogs(j)$: Subset of all blogs $(Examples)$ for which label is j (Conservative or Liberal)

ii) $P(v_j) = Blogs(j)/ Examples.$

iii) $Text(j)$ = A single blog containing all members of $Blogs(j).$

iv) n = Total number of distinct word positions in $Text(j).$

v) For each word w_k in $Vocabulary$

- n_k = number of times word w_k occurs in $Text(j).$
- $P(w_k|v_j) = (n_k + 1)/(n + Vocabulary).$

3.3 Testing

Once we have trained our classifier, the next step is to return the predicted value for the new blog. The following steps are done:

1. $Blog =$ The Blog which needs to be classified.
2. $a_i =$ word found in the i^{th} position within $Blog.$
3. $Positions =$ It has all the word positions in $Blog$ that contain tokens found in $Vocabulary.$
4. Return the label for which $P(v_j) \Pi P(a_i|v_j)$ *where* $i \varepsilon Positions$ is maximum.

For example if the probability of this blog being conservative is more than the probability of this blog being liberal, then we assign the Conservative label to this blog. We keep on repeating this for all Documents in the testing set and we assign a label to the document.

3.4 Calculating Accuracy of the Classifier

Once we get the predicted labels for all the Blogs in the testing set, then we can compare them to their actual label to find the accuracy of the classifier. It is pretty simple to calculate that using the following formula.

$ACC = Percent\ Accuracy.$

$C = Total\ number\ of\ correctly\ classified\ blogs.$

$D = Total\ number\ of\ blogs.$

Then, $ACC = (C/D) * 100$

In the next section, we will describe the results we get using the naive Bayes Approach on the given dataset of blogs.

4.0 Results and Extensions

We split the blog dataset randomly into training set and testing set 9 times to get files split0.train, split0.test and split1.train, split1.test and so on till split8.train, split8.test. We then train our classifier on one training file and evaluate the classifier on the corresponding testing file to find the accuracy of classifier on that testing set. This is repeated 9 times for all sets and we obtain the results as shown in Figure 3.

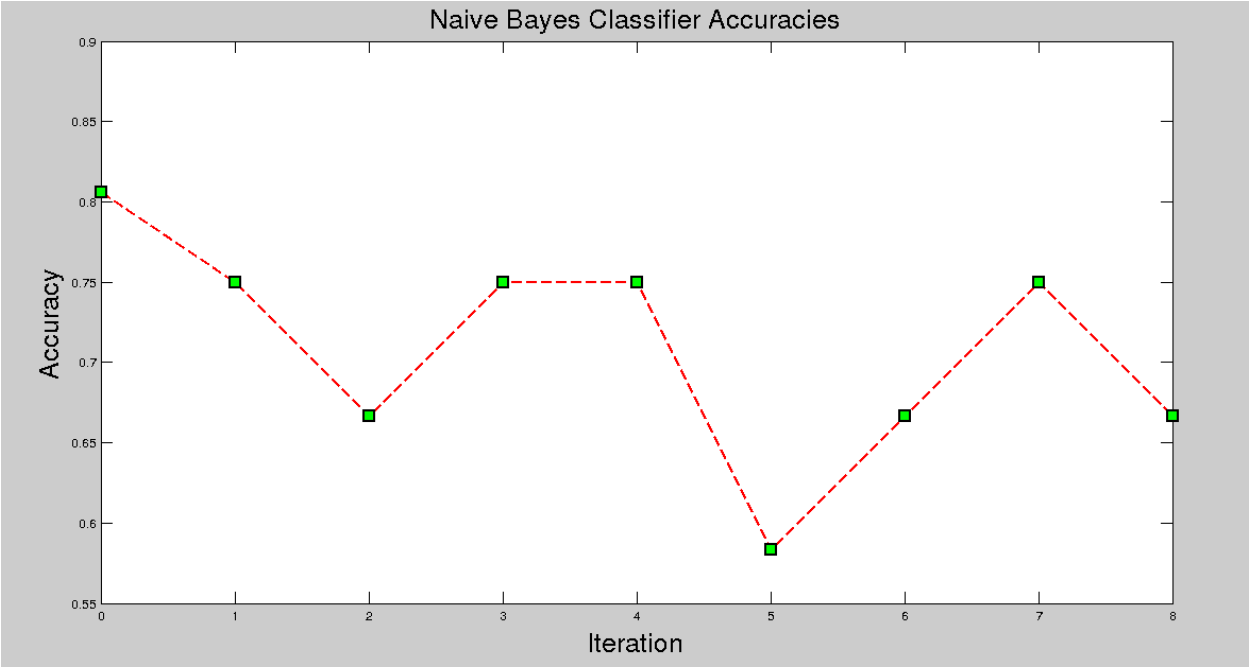

Figure 3. The accuracy of Naive Bayes Classifier for 9 random sets of training and testing blogs

Run **python nb.py split0.train split0.test** on the terminal.

Extension 1

The most common occurring words in the English Language is in the following order: the, be, to, of, and, a, in, that, have, I [2]. These words do not provide us with any relevant information while classifying the blog and therefore can be removed. Another option is to find the top “X” occurring words in the blogs and remove them before applying the Machine Learning algorithm. This leads to the increase in the accuracy of the algorithm. In one set of training and testing data, the top 10 occurring words are given as:
1) In Liberal blogs: the, to, of, and, a, in, that, is, for, on.
2) In Conservative blogs: the, to, of, and, a, in, that, is, for, i.

Once top X occurring words are removed, the accuracy of the algorithm increases.

Run **python nbStopWords.py split0.train split0.test 10** on the terminal to see the result.

Extension 2

One problem is that probabilities can be very close to 0 which can cause problems. We applied smoothing to get rid of this problem, basically the new formula is $P(w_k|v_j) = (n_k + q) / (n + q * |Vocabulary|)$ where q is the smoothing factor.
Run **python smoothing.py split0.train split0.test q** on the terminal to see the result where **q** is the smoothing factor.
Note: the files split0.train and split0.test can be replaced by splitX.train and splitX.test where X = 0,1,2,3,4,5,6,7,8

5.0 Conclusion and Future Work

To conclude, we used the Naive Bayes Classifier to classify the political blogs into Conservative and Liberal blogs and we are able to obtain good accuracy. This has a lot of advantages in case of Big Data when the number of blogs increases and it becomes very difficult to manage them manually.

The future work includes:
1. Although Naive Bayes Classifier is a very good algorithm, there are better and more complex algorithms that can be used to classify large number of blogs more accurately.
2. This can be extended to classify different types and large number of blogs into more than two classes.

6.0 References

1. http://en.wikipedia.org/wiki/Naive_Bayes_classifier
2. http://en.wikipedia.org/wiki/Most_common_words_in_English
3. http://politicalbloglistings.blogspot.com/