# Naïve Bayes: Machine Learning and Text Classification Application of Bayes' Theorem

Owen Baines, University of Leicester

March 2020

# Abstract

This paper introduces the basics of classification and machine learning, as well as building an application of one classification model. The classification model chosen is based on Bayes' Theorem and adapts it to handle large datasets. The paper also discusses the effectiveness of the common F1-Score for statistical analysis as well as also introducing a more meaningful precision-recall measure (Matthews Correlation Coefficient) which is more suited to machine learning algorithms.


# Introduction

## What is Machine Learning?

Algorithms are an essential component for computers to be able to solve and understand problems that we give them. Computers follow human described instructions in order to transform an input to an output. A very simple example of an algorithm is a sorting algorithm, which can be broken down to a simple human process that a computer can follow. There are, however, human processes which do not follow a clearly definable set of instructions – such as species recognition.

As humans, we do not follow an algorithm to identify species, we follow patterns that we have learnt over our lives. Computers have, over time, become more powerful and able to process large datasets and handle increasingly more complex numerical operations – these advancements increase their ability to not only process operations but store and evaluate efficiently. A computer can take a human input of pre-sorted photographs of species, with their associated label, and analyse the pixels to develop its own set of patterns to identify such a species. The process of teaching a computer how to recognise patterns is the basis of Machine Learning.[1]

## Text Classification

Text Classification problems can be solved by using either hand-coded rules or supervised algorithms. Hand-coded rules are purpose built by humans to solve a specific text classification problem. While these can be extremely accurate for simple and clearly defined problems, they do not scale well with more complicated inputs. Further, it can become expensive to maintain rule sets to classify a constantly changing classification problem such as spam emails.

Supervised algorithms handle three inputs (a problem document, defined classes and a pre-classified training set) and output a single classification determined by the algorithm. Compared with hand-coded rules, a supervised algorithm can handle increasingly complicated documents and does not need updating to respond to new trends or changes in classification problems. Further, a supervised algorithm can also be trained to suit an individual's needs such as a spam filter.

# Naïve Bayes

Naïve Bayes is a classification algorithm based on Bayes' Theorem. The algorithm makes two key assumptions about words within documents, these are that:

- probability of a word occurring is independent of the words around it
- all words are conditionally independent with a class

These assumptions are fundamentally simple and would not hold up in the real world where the order of words convey contextual information to a human reader that would not otherwise be conveyed by the same set of words ordered differently. Despite these assumptions, Naïve Bayes has a high degree of accuracy when predicting classes with only a small training set.

## Derivation

We start off by defining Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$$

Where $P(A|B)$ is the conditional probability of event A occurring after event B an $P(B|A)$ is the conditional probability of event B occurring after event A. $P(A)$ and $P(B)$ are the probability of event A and B occurring respectively. $\neg$ is used to denote "not", so $\neg A$ is "not event A".[2]

Assuming we have the following variables associated with the input documents for our classifier:

- $C$ is a set of all classes known ($C = \{c_1, c_2, \dots, c_k\}$)
- $c_i$ is a class in the set $C$
- $N$ is the number of training document inputs
- $N_c$ is the number of training documents with class $c_k$
- $w$ is the set of all words known to the algorithm (a vocabulary; $w = \{w_1, w_2, \dots, w_k\}$)
- $w_k$ is the kth word in a vocabulary
- $n_c$ is the number of times $w_k$ occurs in all documents within class $c_k$
- $n_d$ is the number of all words in all documents within the class
- $n_v$ is the size of set $w$
- $n_t$ is the size of the document being analysed

Looking back at Bayes' Theorem, if we assign events A and B respectively a defined class and a document input such that $P(c_k|w_k)$, we are evaluating the probability of a 'bag' of words being a member of class $c_k$.

We can define the probability of an input document being assigned a class based on the total number of training documents and their existing class assignments:

$$P(c_k) = \frac{N_c}{N}$$

The probability of a word appearing given a certain class can also be defined as:

$$P(w_k|c_i) = \frac{n_c}{n_d + n_v}$$

To complete the derivation, we need to relate these new probability equations back to Bayes' Theorem:

$$P(c_i|w_k) = \frac{P(w_k|c_i)P(c_i)}{P(w_k|c_i)P(c_i) + (1 - P(w_k|c_i))(1 - P(c_i))}$$

This equation introduces an issue if we encounter a new word within a class. As we have no prior probability of $w_k$ in our class, we will have a probability of 0. To prevent this, we can apply a method called 'additive smoothing' where we artificially inflate the value of $n_c$ by one so that the new equation for $P(w_k|c_i)$ is:

$$P(w_k|c_i) = \frac{n_c + 1}{n_d + n_v}$$

To then solve a Naïve Bayes classification problem, we need to find the value of $c_i$ which maximises the probability – we therefore write this as:

$$C_{predicted} = \underset{c \in C}{\operatorname{argmax}} \prod_{k=1}^{n_t} P(c|w_k)$$

With an increasing large document to classify, it may be foreseeable that the probabilities we are multiplying begin to approach the machine precision limit when using floating point arithmetic. To prevent this having an impact when using this method on a computer, we can utilise logs to rewrite the $C_{predicted}$ equation above to be:

$$C_{predicted} = \underset{c \in C}{\operatorname{argmax}} \sum_{k=1}^{n_t} \log\left(P(c|w_k)\right)$$

Worked Example

Using the data available in Appendix 1, we will consider a training set of 5 random sentences and attempt to classify a sentence to show how the algorithm works in practice. For class, 'Accept' labels where a human felt the text provides enough reason to approve a request, while 'Review' was identified as a request requiring more information or further consideration.

| Text | Class |
|---|---|
| provide a platform for an amateur radio community group to share technical material and provide educational material | Accept |
| a university research group wiki for collaboration among the research group and to make the computational aerodynamics research available | Accept |
| I will use primarily for literary creation | Accept |

| | |
|---|---|
| I would love to create a wiki based upon nature it would be a great business | Review |
| I would like to record and share all of my knowledge related with information and technology | Review |

Before we start calculating individual probabilities, we need to define:

- $C = \{Accept, Review\}$
- $N = 5$
- $N_{Accept} = 3$
- $N_{Review} = 2$
- $w = \{$provide, a, platform, for, an, amateur, radio, community, group, to, share, technical, material, and, educational, university, research, wiki, collaboration, among, the, make, computational, aerodynamics, available, I, will, use, primarily, literary, creation, would, love, create, based, upon, nature, it, be, great, business, like, record, all, my, knowledge, related, with, information, technology$\}$
- $n_v = 50$
- $n_d$ for Accept is 43 and for Review is 32.

To improve the efficiency of calculating probabilities for one sentence, we can split the unique words of the new sentence up and only consider those words in each set. For this example, the sentence we are going to classify is, “a wiki to make my university research available”.

The probability of our sentence being classified in either class is:

$$P(Accept) = \frac{3}{5}, P(Review) = \frac{2}{5}$$

Now we need to calculate $P(w_k|c_i)$ for each word and class we are interested in

$$P(a|Accept) = \frac{2+1}{50+43} = \frac{3}{93}, P(a|Review) = \frac{2+1}{50+32} = \frac{3}{82}$$

$$P(wiki|Accept) = \frac{1+1}{50+43} = \frac{2}{93}, P(wiki|Review) = \frac{1+1}{50+32} = \frac{2}{82}$$

$$P(to|Accept) = \frac{1+1}{50+43} = \frac{2}{93}, P(to|Review) = \frac{2+1}{50+32} = \frac{3}{82}$$

$$P(make|Accept) = \frac{1+1}{50+43} = \frac{2}{93}, P(make|Review) = \frac{0+1}{50+32} = \frac{1}{82}$$

$$P(my|Accept) = \frac{0+1}{50+43} = \frac{1}{93}, P(my|Review) = \frac{0+1}{50+32} = \frac{1}{82}$$

$$P(university|Accept) = \frac{1+1}{50+43} = \frac{2}{93}, P(university|Review) = \frac{0+1}{50+32} = \frac{1}{82}$$

$$P(research|Accept) = \frac{2+1}{50+43} = \frac{3}{93}, P(research|Review) = \frac{0+1}{50+32} = \frac{1}{82}$$

$$P(available|Accept) = \frac{1+1}{50+43} = \frac{2}{93}, P(available|Review) = \frac{0+1}{50+32} = \frac{1}{82}$$

Next, we need to calculate the probability of a class given the word being present. For example,

$$P(Accept|a) = \frac{\frac{3}{93} \times \frac{3}{5}}{\left(\frac{3}{93} \times \frac{3}{5}\right) + \left(\frac{90}{93} \times \frac{2}{5}\right)} = \frac{1}{21}$$

$$P(Review|a) = \frac{\frac{3}{82} \times \frac{2}{5}}{\left(\frac{3}{82} \times \frac{2}{5}\right) + \left(\frac{79}{82} \times \frac{3}{5}\right)} = \frac{2}{81}$$

To then classify, we need to solve the $C_{predicted}$ formula from above, substituting in what we have calculated so far:

$$C_{predicted} = \underset{c \in C}{\operatorname{argmax}} \sum_{k=1}^{n_t} \log\left(P(c|w_k)\right)$$

$$C_{Accept} = \sum_{k=1}^{8} \log\left(P(Accept|w_k)\right)$$
$$= \log\left(\frac{1}{21}\right) + \log\left(\frac{3}{94}\right) + \log\left(\frac{3}{94}\right) + \log\left(\frac{3}{94}\right) + \log\left(\frac{3}{187}\right) + \log\left(\frac{3}{94}\right)$$
$$+ \log\left(\frac{1}{21}\right) + \log\left(\frac{3}{94}\right) \approx -11.9192$$

$$C_{Review} = \sum_{k=1}^{8} \log\left(P(Review|w_k)\right)$$
$$= \log\left(\frac{2}{81}\right) + \log\left(\frac{1}{61}\right) + \log\left(\frac{2}{81}\right) + \log\left(\frac{2}{245}\right) + \log\left(\frac{2}{245}\right) + \log\left(\frac{2}{245}\right)$$
$$+ \log\left(\frac{2}{245}\right) + \log\left(\frac{2}{245}\right) \approx -15.4409$$

$$C_{predicted} = \underset{c_i \in C}{\operatorname{argmax}} C_{c_i}$$

Given our largest classification value is produced by using the 'Accept' argument, $C_{predicted} = Accept$. We would then classify the sentence "a wiki to make my university research available" with the label of 'Accept'.

## Accuracy and Precision

For testing the precision of an algorithm, statistics takes a preference to use the F1 score. An F1 score is the harmonic mean between the precision and recall of a statistical test. While an F1 score is a useful evaluator of a test's accuracy and precision, it can be misunderstood easily and does not consider fully the four elements of a confusion matrix. The F1 score favours a test which has a high accuracy of true positives, rather than an accuracy between both true positives and true negatives. An F1 score ranges between 1 and 0, where 1 represents perfect recall and precision.

A confusion matrix is made up of: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

The preferred test for the accuracy of a binary classification algorithm is the Matthews Correlation Coefficient which is a modification of the Pearson Product-Moment Correlation Coefficient. The Matthews Correlation Coefficient as derived by Giuseppe Jurman is:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The possible values range between -1 and +1, where a score of -1 is deemed to be "perfect misclassification" and +1 is "perfect classification".[3] In machine learning, all algorithms strive for a Matthew Correlation Coefficient score of +1 meaning the algorithm classifies perfectly for any theoretical predictive input given.

Assuming we classified a test set of 100 and produced a confusion matrix like the one below:

| | Known Outcome | | |
|---|---|---|---|
| Predicted Outcome | | Accept | Review |
| | Accept | 47 | 11 |
| | Review | 3 | 39 |

We can interpret the confusion matrix as follows:

- True Positives = 47
- True Negatives = 39
- False Positives = 3
- False Negatives = 11

Using this, we can calculate the Matthews Correlation Coefficient as follows:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$
$$= \frac{(47 \times 39) - (3 \times 11)}{\sqrt{(47 + 3)(47 + 11)(39 + 3)(39 + 11)}} = 0.7294$$

Therefore, the above confusion matrix would represent an algorithm which is predicting to a considerable degree of accuracy.

# Conclusion

To evaluate the effectiveness of the Naïve Bayes, we are going to see how the Matthews Correlation Coefficient changes as the training set increases in size. The code used to automate the Naïve Bayes calculations can be found under Appendix 2 and the full data set used can be found under Appendix 1.

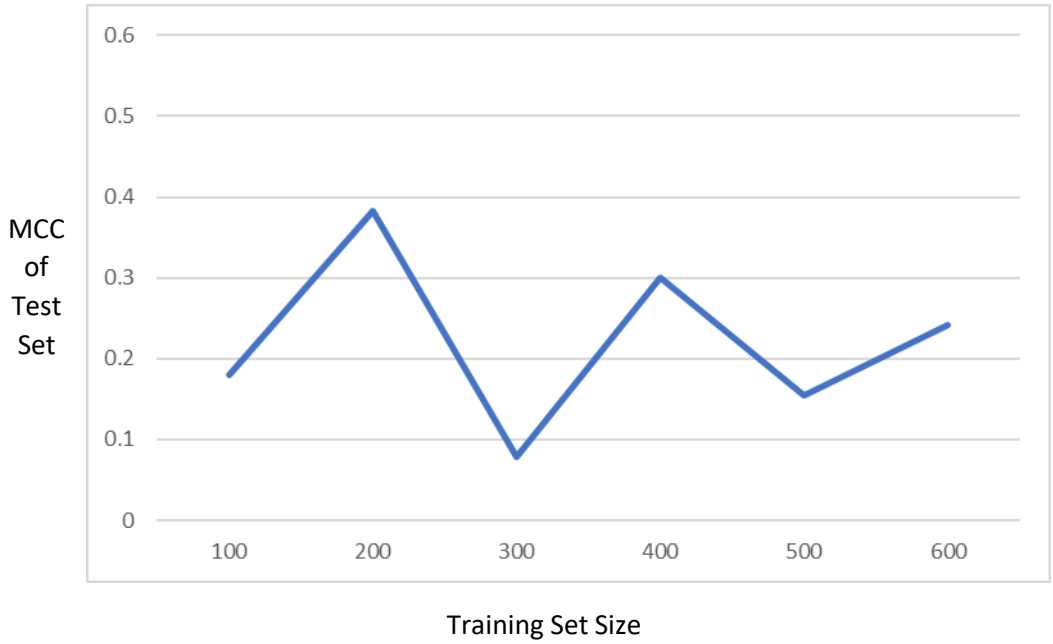


Figure 1 – Matthews Correlation Coefficient for Training Set Sizes

Looking at the figure above, the Naïve Bayes classification algorithm that we have derived does not perform to a high degree of accuracy on the dataset we are applying it to. The scores though do suggest the classification has a weak correlation – meaning it is not random but rather there is some accuracy.

The conclusion we can draw is that Naïve Bayes is too simplistic of a classification algorithm for the application we have investigated in this paper, and more complex methods such as artificial neural networks may be better suited to the understanding of how humans interpret and classify certain sentences with respect to whether they convey enough information to be able to draw intelligent conclusions from.

Although Naïve Bayes seems too simplistic here, it would perform effectively in classifying in sentimental analysis where words convey a clearly defined and unambiguous meaning such as positive and negative reviews on websites. This paper provides a derivation of Bayes' Theorem that can be used for such an analysis and provides modifications to the theorem to account for real world problems such as new data and exponentially large datasets.

# References

[1] Alpaydin, E. (2010). *Introduction to Machine Learning*. Cambridge, Mass.: MIT Press, pp.1-2.

[2] Cichosz, P. (2015). *Data mining algorithms.* John Wiley & Sons, Incorporated pp. 118-126.

[3] Chicco, D., Jurman, G. *The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation.* BMC Genomics 21, 6 (2020).

# Appendices

Appendix 1: https://github.com/OwenBaines/TextClassify/blob/master/res/Miraheze-Requests-Comments-Result.csv

Appendix 2: https://github.com/OwenBaines/TextClassify/blob/master/res/naiiveBayes.m