**Methodology**

DATA COLLECTION

INITIALISATION

PREPROCESSING OF DATA

STEP

ATTRIBUTE SELECTION (BAG OF WORDS)

ALGORITHM

TRAINING MODEL

LEARNING STEP

TEST SET

EVALUATION STEP

**NAÏVE BAYES CLASSIFICATION**

It is a [classification technique](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2/?utm_source=blog&utm_medium=6stepsnaivebayesarticle) based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

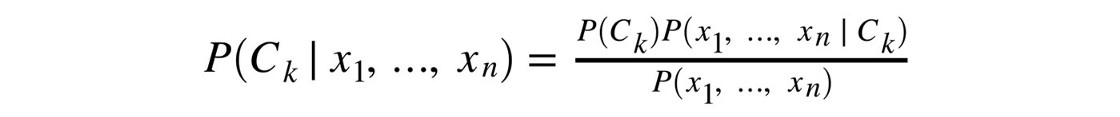
Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

## 4 Applications of Naive Bayes Algorithms

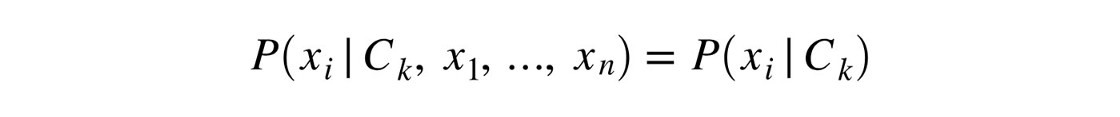
* **Real time Prediction**
* **Multi class Prediction**
* **Text classification/ Spam Filtering/ Sentiment Analysis**
* **Recommendation System**

**Multinomial Naive Bayes** classification algorithm tends to be **a baseline solution** for sentiment analysis task. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes.

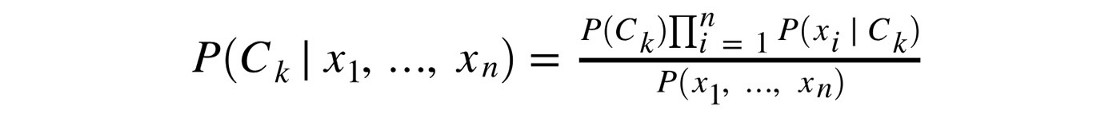
Given the dependent feature vector **(x₁, …, xn)** and the class **Ck**. Bayes’ theorem is stated mathematically as the following relationship:



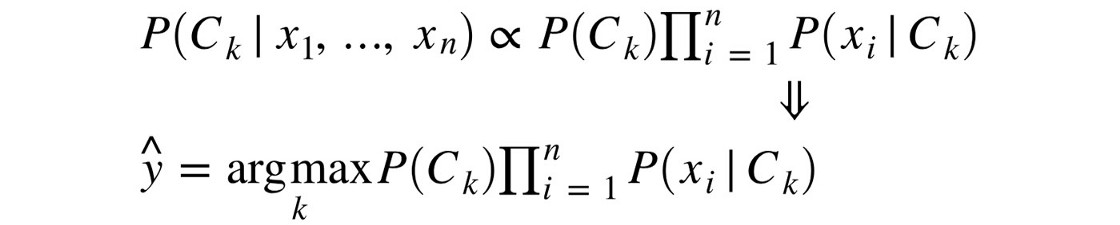
According to the “naive” conditional independence assumptions, for the given class **Ck** each feature of vector **xi**is conditionally independent of every other feature **xj** for **i≠j**..



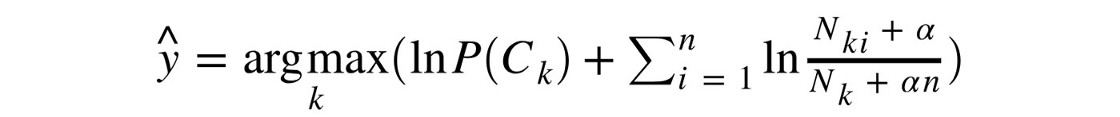
Thus, the relation can be simplified to



Since **P(x₁, …, xn)** is constant, if the values of the feature variables are known, the following classification rule can be used:



Thus, the final decision rule is defined as follows:



# Loading sentiment data

The files positive.csv and negative.csv contain positively labelled and negatively labelled tweets, respectively.

# Preprocessing data

Texts generated by humans in social media sites contain lots of noise that can significantly affect the results of the sentiment classification process. Moreover, depending on the features generation approach, every new term seems to add at least one new dimension to the feature space. That makes the feature space more sparse and high-dimensional. Consequently, the task of the classifier has become more complex.

To prepare messages, such text preprocessing techniques as replacing URLs and usernames with keywords, removing punctuation marks and converting to lowercase were used in this program.

**Training Multinomial Naive Bayess**

A Pipeline class was used to make the vectorizer => transformer => classifier easier to work with. Such hyper-parameters as n-grams range, IDF usage, TF-IDF normalization type and Naive Bayes alpha were tunned using grid search. The performance of the selected hyper-parameters was measured on a test set that was not used during the model training step.

The dataset was splitted into train and test subsets.

As we could see, even a very basic implementation of the Naive Bayes algorithm can lead to surprisingly good results for the task of sentiment analysis. This model is essentially a binary classifier, meaning that it can be applied to any dataset in which we have two categories. There are all kinds of applications for it, ranging from spam detection to bitcoin trading based on sentiment. With an accuracy of 82%, there is really a lot that you could do, all you need is a labeled dataset and of course, the larger it is, the better!