**Naive Bayes Classifier**

# ****What category of algorithms does the Naive Bayes classifier belong to?****

Naive Bayes classifier is based on the Bayes’ Theorem, adapted for use across different machine learning problems. These include **classification**, **clustering**, and **network analysis**. This story will explain how Naive Bayes is used for **classification** problems that sit under the supervised branch of the Machine Learning tree.

Talking about supervised learning, a quick reminder of the difference between regression and classification:

* **Regression** aims to predict the value of a continuous target variable (e.g., price of a house)
* **Classification** aims to predict the class label of a categorical target variable (e.g., spam email / not-spam email)

# ****How does the Naive Bayes classifier work?****

Let’s start by answering the following question first.

## **Why is Naive Bayes naive?**

Naive Bayes' underlying assumption is that the predictors (attributes / independent variables) **are independent**of each other. This is a big assumption because it is easy to show that there is often at least some correlation between variables in real life. It is precisely this assumption of independence that makes Bayes classification “naive.”

Nevertheless, the Naive Bayes algorithm has been shown time and time again to perform really well in classification problems, despite the assumption of independence. Simultaneously, it is a fast algorithm since it scales easily to include many predictors without having to handle multi-dimensional correlations.

## **Conditional probabilities**

To understand Naive Bayes, we first need to understand conditional probabilities. For that, let’s use the below example.

Assume we have a bucket filled with red and black balls. In total, there are 15 balls: 7 red and 8 black.

The probability of randomly picking a red ball out of the bucket is 7/15. You can write it as P(red) = 7/15.

If we were to draw balls one at a time without replacing them, what is the probability of getting a black ball on a second attempt after drawing a red one on the first attempt?

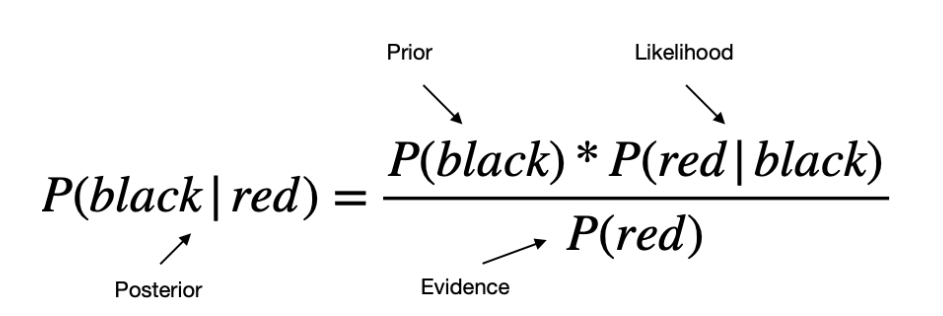
You can see that the above question is worded to provide us with the condition that needs to be satisfied first before the second attempt is made. That condition says that a red ball must be drawn during the first attempt.

As stated earlier, the probability of getting a red ball on the first attempt (P(red)) is 7/15. That leaves 14 balls inside a bucket with 6 red and 8 black. Hence, the probability of getting a black ball next is 8/14 = 4/7.

We can write this as a **conditional probability**:  
**P(black|red) = 4/7.** *(read: probability of black given red)*We can also see that   
P(red and black) = P(red) \* P(black|red) = 7/15 \* 8/14 = 4/15.Similarly,  
P(black and red) = P(black) \* P(red|black) = 8/15 \* 7/14 = 4/15.

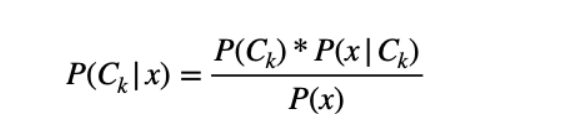
## **Bayes’ theorem**

The Bayes’ theorem helps us calculate conditional probabilities of an event when we know the likelihood of a reverse event. Using the example above, we would write it as follows:



## **Naive Bayes classifier**

Let’s now take the above equation and change the notation to make it more relevant for classification problems.

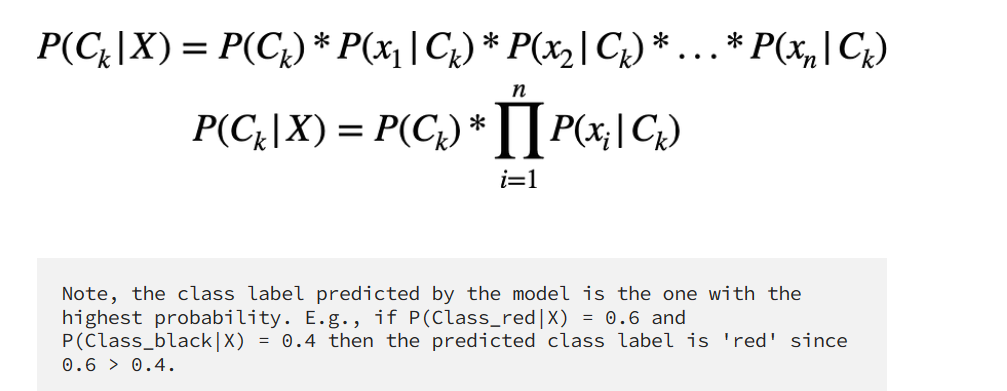


where:

* **P(C|x)**is the posterior probability of class C (target variable) given the predictor x (attribute / independent variable);
* **P(C)** is the prior probability of class C;
* **P(x|C)** is the likelihood, which is the probability of the predictor x given class C;
* **P(x)** is the prior probability of the predictor x;
* **Little k**is just the notation to distinguish between different classes as you would have at least 2 separate classes in the classification scenario (e.g., spam / not-spam, red ball / black ball).

In practice, there is interest only in the numerator of the above equation since the denominator does not depend on C. Also, since all of the values of the attributes x are known, the denominator is effectively a constant.

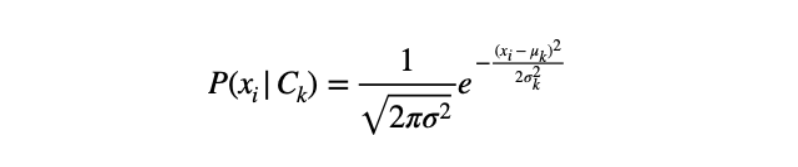
So, combining the above with the assumption of independence and taking into account multiple predictors, the equation for classification becomes:



## **Gaussian Naive Bayes — adaptation for continuous attributes**

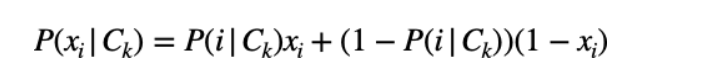
When dealing with continuous data, a typical assumption is that each class's continuous values are distributed according to a normal (a.k.a. Gaussian) distribution.

While we can use frequencies to calculate probabilities of occurrence for categorical attributes, we cannot use the same approach for continuous attributes. Instead, we first need to calculate the mean and variance for x in each class and then calculate P(x|C) using the following formula:



## **Bernoulli Naive Bayes — adaptation for boolean attributes**

If you have binary-valued attributes (Bernoulli, boolean), then you can use a Bernoulli NB model which utilizes the following formula for calculation of P(x|C):



The important part is identifying which Naive Bayes' variation to use given the type of attributes (independent variables) you have:

1. **Gaussian NB** with 2 independent variables  
   2. **Gaussian NB with 3 class labels** and 2 independent variables  
   3. **Categorical NB** with 2 independent variables  
   4. **Bernoulli NB** with 1 independent variable  
   5. **Mixed NB**(Gaussian + Categorical) **approach 1** — convert continuous variables into categorical ones through binning and then train a categorical model
2. **MultinomialNB()**