Scientific Machine Learning 11
Naïve Bayes Classifier

Donghyun Ko

January 4, 2026

What you will learn in this post.

- How Bayes' theorem connects prior, likelihood, and posterior probabilities.
- What conditional independence means and why Naïve Bayes assumes it.
- How to derive the Naïve Bayes classifier and the MAP (maximum a posteriori) rule.
- Differences among the main event models: Gaussian, Multinomial, and Bernoulli.
- Typical applications (spam filtering, sentiment analysis, document classification) and strengths/weaknesses of Naïve Bayes.

Contents

1 Introduction

Naïve Bayes (NB) is a family of supervised learning algorithms grounded in **Bayes' theorem** but simplified by a bold assumption: *all input features are conditionally independent given the class label.* Despite this simplification, NB performs remarkably well in practice—particularly in text and document classification. Before diving in, recall that:

- *K-Nearest Neighbors (KNN)*—a non-parametric, distance-based method.
- This posting introduces a probabilistic, parametric alternative that models the likelihood of features given each class.

NB is favored for its simplicity, interpretability, and computational efficiency. It often achieves strong results even on large-scale data where iterative optimization (like in logistic regression) would be costly.

2 Mathematical Foundations

2.1 Bayes' Theorem

The Bayes' theorem expresses how we update beliefs given new evidence:

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

where:

- A and B are different events, and $P(B) \neq 0$,
- $P(A)$: prior belief about event A,
- $P(B|A)$: likelihood of A given a fixed B – how probable B is given A,
- $P(B)$: evidence (a normalizing constant) – marginal probability of event B
- $P(A|B)$: posterior probability of A after seeing B.

2.2 Conditional Independence

NB's "naïve" assumption is that the features are independent once the class label is known:

$$P(X_i|X_{\sim i}, Y) = P(X_i|Y), \quad i = 1, \ldots, n$$

This assumption drastically simplifies the joint probability and allows efficient computation even with many features.

Example: It is reasonable to assert that Thunder (X) is independent of Rain (Y) *given* Lightning (Z). Since Lightning causes Thunder, once we know whether there is Lightning or not, additional information about Rain provides no extra knowledge about Thunder. *Of course, there is a clear dependence of Thunder on Rain in general, but there is no conditional*

dependence once we know the value of Lightning. Although X, Y and Z are each single random variables in this example, more generally the definition applies to sets of random variables. For example, we might assert that variables A,B are conditionally independent of C,D given variables E,F.

2.3 From Bayes' Rule to Naïve Bayes

Naïve Bayes (NB) starts from Bayes' theorem, which decomposes the conditional probability of a class y_k given a feature vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ as

$$P(y_k|\mathbf{X}) = \frac{P(\mathbf{X}|y_k)\, P(y_k)}{P(\mathbf{X})}$$

Here,

- $P(y_k)$ is the **prior probability** of class y_k,
- $P(\mathbf{X}|y_k)$ is the **likelihood** of observing features $\mathbf{X}$ given class y_k,
- $P(\mathbf{X})$ is the **marginal probability (evidence)**, a constant across all classes.

Because $P(\mathbf{X})$ does not depend on y_k, it is usually not of direct interest when comparing classes—it serves only as a normalization constant. The numerator

$$P(\mathbf{X}|y_k)P(y_k) = P(\mathbf{X}, y_k) = P(X_1, X_2, \ldots, X_n, y_k)$$

represents the **joint probability model**. Using the **chain rule of probability**, this can be expanded as.

$$\begin{aligned} P(X_1, X_2, \ldots, X_n, y_k) &= P(X_1|X_2, \ldots, X_n, y_k)\, P(X_2, \ldots, X_n, y_k) \\ &= P(X_1|X_2, \ldots, X_n, y_k)\, P(X_2|X_3, \ldots, X_n, y_k)\, P(X_3, \ldots, X_n, y_k) \\ &= P(X_1|X_2, \ldots, X_n, y_k) \cdots P(X_{n-1}|X_n, y_k)\, P(X_n|y_k)\, P(y_k) \end{aligned}$$

At this stage, computing all these conditional terms is infeasible for large n. The **naïve conditional independence assumption** simplifies the model drastically:

Naïve conditional independence assumption: All features X_i are independent of each other *given* the class label y_k:

$$P(X_i \mid \mathbf{X}_{\sim i}, y_k) = P(X_i \mid y_k), \qquad i = 1, 2, \ldots, n,$$

where $\mathbf{X}_{\sim i} = \{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\}$. Changing one feature does not affect any other features once the class is fixed.

Applying this assumption reduces the joint probability to a product of single-feature likelihoods such that:

$$P(X_1, X_2, \ldots, X_n, y_k) = P(y_k) \prod_{i=1}^{n} P(X_i|y_k)$$

Substituting into the Bayes' rule gives the simplified **Naïve Bayes model**:

$$P(y_k|\mathbf{X}) = \frac{P(\mathbf{X}|y_k)P(y_k)}{P(\mathbf{X})} \propto P(y_k)\prod_{i=1}^{n} P(X_i|y_k)$$

where $\propto$ denotes proportionality since $P(\mathbf{X})$ is constant. This final expression is called the **independent feature model** or the **Naïve Bayes probability model**. It forms the foundation for all event models (Gaussian, Multinomial, Bernoulli) discussed next.

2.4 MAP Decision Rule

We choose the most probable class:

$$\hat{Y} = \underset{k\in\{1,\ldots,K\}}{\arg\max}\, P(y_k)\prod_{i=1}^{n} P(X_i|y_k)$$

- $P(y_k)$: prior—either uniform $(1/K)$ or estimated from frequency in the training set.
- $P(X_i|y_k)$: likelihood of the features' distributions—one must assume a distribution or generate nonparametric models for the features from the training set; $P(X_i|y_k)$ is then estimated using an appropriate distribution (The assumptions on $P(X_i|y_k)$ are called the *event model* of the NB classifier). The different NB classifiers differ mainly by the assumptions they make regarding on $P(X_i|y_k)$.

Naïve Bayes assigns a class label that makes the observed features most likely under the assumed independence model. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori (MAP) or the MAP decision rule. So, this Bayes classifier is the function that assigns a class label $\hat{Y} = y_k$ for some $k \in \{1, 2, ...K\}$.

2.5 Gaussian Naïve Bayes (Continuous Features)

When dealing with **continuous data**, we typically assume that the feature values in each class follow a **Gaussian (normal) distribution**. This leads to the so-called *Gaussian Naïve Bayes* model. Formally, for each feature X_i and class y_k:

$$P(X_i \mid y_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left[-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right]$$

Here, the parameters $\mu_{k,i}$ and $\sigma_{k,i}^2$ represent the mean and variance of feature i within class y_k, estimated from the training data. The estimation process is simple and analytical—no iterative optimization is required.

- First, partition the training data by class label y_k.
- For each class y_k, compute the sample mean and the unbiased (Bessel-corrected) variance:

$$\mu_{k,i} = \frac{1}{N_k}\sum_{j=1}^{N_k} x_{j,i}, \qquad \sigma_{k,i}^2 = \frac{1}{N_k - 1}\sum_{j=1}^{N_k}(x_{j,i} - \mu_{k,i})^2,$$

  where N_k is the number of samples belonging to class y_k.

- Each feature's likelihood given class y_k is then modeled as the Gaussian density in:

$$P(X_i \mid y_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left[-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right]$$

Interpretation. For a given sample $\mathbf{x} = (x_1, \ldots, x_n)$, the overall class likelihood is

$$P(\mathbf{x} \mid y_k) = \prod_{i=1}^{n} P(X_i = x_i \mid y_k),$$

and the posterior is

$$P(y_k \mid \mathbf{x}) \propto P(y_k) \prod_{i=1}^{n} P(X_i \mid y_k).$$

Thus, each feature contributes multiplicatively to the total likelihood.

Example (One ontinuous feature with two classes). Suppose that there are two classes y_0, y_1 with priors $P(y_0) = 0.6$, $P(y_1) = 0.4$. Assume:

$$X \mid y_0 \sim \mathcal{N}(\mu_0{=}0, \sigma^2{=}1), \qquad X \mid y_1 \sim \mathcal{N}(\mu_1{=}2, \sigma^2{=}1)$$

For a new observation x, we compare their log-posteriors (constants in the denominator canceled out):

$$\log P(y_1|x) - \log P(y_0|x) = \log\frac{P(y_1)}{P(y_0)} - \frac{(x-\mu_1)^2 - (x-\mu_0)^2}{2}$$

Substituting the given numbers:

$$\Delta(x) = \log\frac{0.4}{0.6} - \tfrac{1}{2}\Big[(x-2)^2 - x^2\Big] = \log\frac{2}{3} + 2x - 2$$

Setting $\Delta(x) = 0$ gives the **decision boundary**:

$$x^\star = 1 - \tfrac{1}{2}\log\frac{2}{3} = 1 + \tfrac{1}{2}\log\frac{3}{2} \approx 1.203$$

Classification rule: predict class y_1 if $x > 1.203$, otherwise y_0.

Key takeaway. Gaussian NB assumes each class generates its features from a normal distribution with its own mean and variance. Decision boundaries are determined by comparing class-specific densities and priors, yielding quadratic or linear discriminants depending on whether variances are equal.

2.6 Multinomial Naïve Bayes (Count or Frequency Data)

When features represent discrete counts or frequencies (e.g., word occurrences in a document), the **Multinomial Naïve Bayes** (MNB) model is appropriate. Each feature corresponds to an event that may occur multiple times, and the feature vector can be viewed as a histogram:

$$\mathbf{X} = (X_1, X_2, \ldots, X_n), \quad X_i = x_i = \text{number of times event } i \text{ occurs.}$$

Under a **multinomial event model**, the likelihood of observing a specific histogram $\mathbf{X} = \mathbf{x}$ given class y_k is

$$P(\mathbf{X} = \mathbf{x} \mid y_k) = \frac{\left(\sum_{i=1}^{n} x_i\right)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p_{ki}^{x_i},$$

where:

- p_{ki} is the probability that event i occurs within class y_k, and
- $\sum_{i=1}^{n} x_i$ is the total number of observed events (tokens) in the instance.

Why the multinomial likelihood has this form. Fix a class y_k. Under the *multinomial event model*, an instance with total length $N = \sum_{i=1}^{n} x_i$ is generated by N i.i.d. draws from a categorical distribution over n events with parameters $\mathbf{p}_k = (p_{k1}, \ldots, p_{kn})$, $\sum_i p_{ki} = 1$.

Step 1: Probability of one concrete sequence. Consider any specific ordered sequence of N draws whose histogram equals $\mathbf{x} = (x_1, \ldots, x_n)$ (i.e., event i appears exactly x_i times). By independence,

$$P(\text{that sequence} \mid y_k) = \prod_{i=1}^{n} p_{ki}^{x_i}.$$

(Each time event i appears, it contributes a factor p_{ki}; multiplying over all occurrences yields $p_{ki}^{x_i}$, and multiplying over i gives the product.)

Step 2: How many such sequences realize the same histogram? All orderings that place the x_1 copies of event 1, the x_2 copies of event 2, …, and the x_n copies of event n are distinct sequences but share the same counts $\mathbf{x}$. The number of distinct orderings is the *multinomial coefficient*

$$\binom{N}{x_1, x_2, \ldots, x_n} = \frac{N!}{x_1!\, x_2! \cdots x_n!}$$

Step 3: Sum over all sequences with the same histogram. The likelihood of observing *the histogram* $\mathbf{X} = \mathbf{x}$ is the sum of the probabilities of all sequences that yield $\mathbf{x}$. Since each such sequence has the same probability $\prod_i p_{ki}^{x_i}$ (Step 1), and there are $\binom{N}{x_1,\ldots,x_n}$ of them (Step 2),

$$\begin{aligned} P(\mathbf{X} = \mathbf{x} \mid y_k) &= \binom{N}{x_1, \ldots, x_n} \prod_{i=1}^{n} p_{ki}^{x_i} \\ &= \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} p_{ki}^{x_i} \end{aligned}$$

Log-likelihood (useful in computation) is

$$\log P(\mathbf{x} \mid y_k) = \log N! - \sum_i \log x_i! + \sum_i x_i \log p_{ki},$$

where the first two terms cancel between classes in MAP comparisons.

Interpretation. The multinomial coefficient $\frac{(\sum_i x_i)!}{\prod_i x_i!}$ comes from combinatorics—it counts how many distinct orderings of the same histogram exist. It generalizes the binomial coefficient from two outcomes ($m = 2$) to m possible outcomes:

$$(a_1 + a_2 + \cdots + a_m)^n = \sum_{k_1+k_2+\cdots+k_m=n} \frac{n!}{k_1!k_2!\cdots k_m!} \prod_{i=1}^{m} a_i^{k_i}. \tag{13–14}$$

In document classification, a_i corresponds to the probability of each word, and k_i is the count of that word in the document.

A practical issue arises when a particular word or feature never occurs in a class during training ($x_i = 0$): its estimated probability $p_{ki} = 0$ makes the entire product zero. To prevent this, we apply **Laplace or Lidstone smoothing**, also known as adding a *pseudocount*:

$$\boxed{\hat{p}_{ki} = \frac{n_{ki} + \alpha}{N_k + \alpha V},}$$

where

- n_{ki} is the observed count of event i in class y_k,
- $N_k = \sum_i n_{ki}$ is the total count of all events in class y_k,
- V is the vocabulary (feature) size,
- $\alpha > 0$ is the pseudocount ($\alpha = 1$ for Laplace, $\alpha < 1$ for Lidstone smoothing).

Why smoothing? Without it, any unseen word makes $P(\mathbf{X} \mid y_k) = 0$. Smoothing distributes a small nonzero probability mass across all words, ensuring stable likelihood estimation even for rare or unseen events.

Example with Laplace Smoothing. Consider a vocabulary {`good`, `bad`, `great`} and two classes (`pos`, `neg`) with equal priors $P(\text{pos}) = P(\text{neg}) = 0.5$. Training word counts are summarized below:

	`good`	`bad`	`great`	total
pos	3	1	2	6
neg	1	3	0	4

Applying Laplace smoothing ($\alpha = 1$) with $V = 3$:

$$p(\texttt{good}|\text{pos}) = \frac{3+\alpha}{6+V\alpha} = \frac{4}{9}, \quad p(\texttt{bad}|\text{pos}) = \frac{1+\alpha}{9} = \frac{2}{9}, \quad p(\texttt{great}|\text{pos}) = \frac{2+\alpha}{9} = \frac{3}{9};$$
$$p(\texttt{good}|\text{neg}) = \frac{1+\alpha}{4+3} = \frac{2}{7}, \quad p(\texttt{bad}|\text{neg}) = \frac{3+\alpha}{7} = \frac{4}{7}, \quad p(\texttt{great}|\text{neg}) = \frac{0+\alpha}{7} = \frac{1}{7}.$$

For a new document $\mathbf{x} = (1, 0, 1)$ (`good`=1, `bad`=0, `great`=1):

$$\begin{aligned} \text{pos:} \quad & (4/9) \times (3/9) = \tfrac{4}{27} \approx 0.1481 \\ \text{neg:} \quad & (2/7) \times (1/7) = \tfrac{2}{49} \approx 0.0408 \end{aligned}$$

Thus,

$$P(\text{pos})P(\mathbf{x}|\text{pos}) > P(\text{neg})P(\mathbf{x}|\text{neg}),$$

and the model predicts the class **positive**.

Summary. Multinomial NB models count-based features (e.g., words, events) and uses Laplace/Lidstone smoothing to prevent zero probabilities. It is the most common Naïve Bayes variant in *text classification, spam detection, and topic labeling.*

2.7 Bernoulli Naïve Bayes (Binary Features)

When each feature is binary ($x_i \in \{0, 1\}$), indicating the *presence or absence* of an event, the Naïve Bayes likelihood follows a **Bernoulli distribution**.

Step 1: Single-variable Bernoulli distribution. A Bernoulli random variable Y takes value 1 with probability p and 0 with probability $1 - p$:

$$f(y; p) = \begin{cases} p, & y = 1, \\ 1 - p, & y = 0. \end{cases}$$

This can be written compactly as

$$f(y; p) = p^y (1 - p)^{1-y},$$

because when $y = 1$ the term $(1 - p)^{1-y}$ equals 1, and when $y = 0$ the term p^y equals 1. Hence, the expression automatically selects the correct branch without using an explicit piecewise definition.

Step 2: Extending to multiple independent binary features. Assume n binary features $X_1, \ldots, X_n$, each following a Bernoulli distribution conditioned on class y_k with success probability $p_{ki} = P(X_i = 1 \mid y_k)$. Under the independence assumption,

$$P(\mathbf{X} = \mathbf{x} \mid y_k) = \prod_{i=1}^{n} f(x_i; p_{ki}) = \prod_{i=1}^{n} p_{ki}^{x_i} (1 - p_{ki})^{(1-x_i)}$$

Each factor contributes p_{ki} if feature i is present ($x_i = 1$) and $(1-p_{ki})$ if absent ($x_i = 0$). Thus, the model captures binary occurrence rather than frequency—appropriate for applications where only whether a feature appears matters (e.g., "word appears / does not appear" in document classification).

Intuition. The Bernoulli NB model assumes each feature independently flips a biased coin with success probability p_{ki}. The overall likelihood multiplies these independent contributions. Compared with Multinomial NB (which counts frequencies), Bernoulli NB focuses on binary occurrence information.

Example (binary presence/absence). Binary features: $\{\texttt{offer}, \texttt{win}, \texttt{meeting}\}$. Class priors: $P(\text{spam}) = 0.4$, $P(\text{ham}) = 0.6$. Estimated conditional probabilities:

$$\begin{aligned}&\text{spam: } p(\texttt{offer}|s) = 0.8,\ p(\texttt{win}|s) = 0.7,\ p(\texttt{meeting}|s) = 0.1;\\ &\text{ham: } p(\texttt{offer}|h) = 0.1,\ p(\texttt{win}|h) = 0.05,\ p(\texttt{meeting}|h) = 0.6.\end{aligned}$$

For an email with $\mathbf{x} = (1, 1, 0)$ (contains `offer` and `win`, not `meeting`):

$$\begin{aligned}P(\mathbf{x}|\text{spam}) &= 0.8 \times 0.7 \times (1 - 0.1) = 0.504\\ P(\mathbf{x}|\text{ham}) &= 0.1 \times 0.05 \times (1 - 0.6) = 0.002\end{aligned}$$

Posterior (unnormalized):

$$P(\text{spam})P(\mathbf{x}|\text{spam}) = 0.4 \times 0.504 = 0.2016, \quad P(\text{ham})P(\mathbf{x}|\text{ham}) = 0.6 \times 0.002 = 0.0012.$$

Decision rule (MAP): predict spam.

Key takeaway. Bernoulli NB models binary "yes/no" features—ideal when only the occurrence, not frequency, of a feature carries information. Typical use cases: document classification, email spam detection, and sentiment analysis.

3 Applications and Discussions

3.1 Why Naïve Bayes Works Surprisingly Well

Although the independence assumption rarely holds exactly, Naïve Bayes often achieves strong performance because:

- It needs *few training samples*—just enough to estimate simple per-feature distributions.
- It is *robust to irrelevant features*; each contributes independently.
- It is *computationally efficient*—no gradient descent or matrix inversion.
- The decoupled one-dimensional feature distributions help reduce the "curse of dimensionality."

3.2 Typical Use-Cases

- **Spam filtering:** classify emails as "spam" or "not spam" based on word occurrences.
- **Sentiment analysis:** predict whether a review expresses positive or negative emotion.
- **Document classification:** assign categories (sports, politics, tech, etc.) to news articles.
- **Recommendation and tagging:** use NB to model user preferences or tag probability.

3.3 Strengths and Limitations

- **Pros:** fast, simple, interpretable, strong baseline for many NLP tasks.
- **Cons:** assumes independence (often false), produces poor probability calibration (good classifier, bad estimator).

In practice, NB is a great first model—easy to implement and surprisingly competitive, especially when features are relatively independent or datasets are high-dimensional but sparse.

This is a posting that I summarized for study purposes and adapted from lecture notes of NE-795 (Scientific Machine Learning) given by Professor Xu Wu, NC State University.