Naive Bayes Classification — A Worked Tutorial

This tutorial explains the naive Bayesian classifier using the well-known *buys_computer* training set. We build up the idea from Bayes' theorem, then classify a new customer step by step, and finally handle the problem of zero-count probabilities.

1. What the classifier is trying to do

We have a customer described by four attributes and we want to predict whether they will buy a computer. Each customer is a tuple

$$\mathbf{X} = (x_1, x_2, x_3, x_4) = (\text{age},\ \text{income},\ \text{student},\ \text{credit\_rating}),$$

and the class label *buys_computer* takes two values, **yes** or **no**. Given a new customer's attributes, we want the class with the highest probability — the *most probable* label given the evidence.

2. Bayes' theorem, briefly

For a class C_i and evidence $\mathbf{X}$, Bayes' theorem gives the posterior probability

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)\, P(C_i)}{P(\mathbf{X})}.$$

Here $P(C_i)$ is the **prior** (how common the class is overall), $P(\mathbf{X} \mid C_i)$ is the **likelihood** (how typical this evidence is within the class), and $P(C_i \mid \mathbf{X})$ is the **posterior** we want. Because $P(\mathbf{X})$ is the same for every class, we can ignore it when comparing classes and simply choose the class that maximizes

$$P(\mathbf{X} \mid C_i)\, P(C_i).$$

3. The "naive" assumption

The difficulty is $P(\mathbf{X} \mid C_i)$: estimating the joint behaviour of all attributes together needs enormous data. The naive Bayesian classifier sidesteps this by assuming **class-conditional independence** — that, once the class is known, the attributes do not influence one another. Under this assumption the joint likelihood factorizes into a simple product:

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times \cdots \times P(x_n \mid C_i).$$

For a categorical attribute, each factor $P(x_k \mid C_i)$ is estimated by counting: the number of training tuples in class C_i that have the value x_k, divided by the total number of tuples in class C_i.

4. The training data

The training set has 14 customers. Let C_1 = (*buys_computer* = yes) and C_2 = (*buys_computer* = no).

RID	age	income	student	credit_rating	buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Counting the class labels: 9 customers bought a computer and 5 did not.

5. The customer we want to classify

We wish to predict the label of

$$\mathbf{X} = (\text{age} = \text{youth},\ \text{income} = \text{medium},\ \text{student} = \text{yes},\ \text{credit\_rating} = \text{fair}).$$

Step 1 — Prior probabilities

$$P(\text{buys\_computer} = \text{yes}) = \frac{9}{14} = 0.643, \qquad P(\text{buys\_computer} = \text{no}) = \frac{5}{14} = 0.357.$$

Step 2 — Conditional probabilities for each attribute value

Counting each attribute value within each class gives:

conditional probability	value
$P(\text{age} = \text{youth} \mid \text{yes})$	$2/9 = 0.222$
$P(\text{age} = \text{youth} \mid \text{no})$	$3/5 = 0.600$
$P(\text{income} = \text{medium} \mid \text{yes})$	$4/9 = 0.444$
$P(\text{income} = \text{medium} \mid \text{no})$	$2/5 = 0.400$
$P(\text{student} = \text{yes} \mid \text{yes})$	$6/9 = 0.667$
$P(\text{student} = \text{yes} \mid \text{no})$	$1/5 = 0.200$
$P(\text{credit\_rating} = \text{fair} \mid \text{yes})$	$6/9 = 0.667$
$P(\text{credit\_rating} = \text{fair} \mid \text{no})$	$2/5 = 0.400$

Step 3 — Likelihood of the evidence for each class

Multiply the four conditional probabilities within each class:

$$P(\mathbf{X} \mid \text{yes}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044,$$

$$P(\mathbf{X} \mid \text{no}) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

Step 4 — Combine with the priors and compare

$$P(\mathbf{X} \mid \text{yes})\, P(\text{yes}) = 0.044 \times 0.643 = 0.028,$$

$$P(\mathbf{X} \mid \text{no})\, P(\text{no}) = 0.019 \times 0.357 = 0.007.$$

Since $0.028 > 0.007$, the naive Bayesian classifier predicts **buys_computer = yes** for this customer.

6. Continuous attributes (a note)

If an attribute is continuous rather than categorical, we do not count. Instead we usually assume the values follow a normal (Gaussian) distribution,

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

and estimate the mean μ_{C_i} and standard deviation σ_{C_i} of that attribute from the training tuples of class C_i. The conditional probability is then $P(x_k \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$. Everything else in the procedure stays the same.

7. The zero-probability problem and the Laplacian correction

There is a weakness in the counting scheme. Because the likelihood is a **product**, a single conditional probability of 0 makes the entire product 0 — one unseen attribute value cancels all the other evidence, no matter how strong it is. For example, if no training customer in some class was a student, then $P(\text{student} = \text{yes} \mid \text{that class}) = 0$ and that class is ruled out entirely.

The fix, called the **Laplacian correction** (or Laplace estimator), is to add 1 to each count and adjust the denominator accordingly, so no probability is ever exactly zero.

Illustration. Suppose for class *buys_computer* = yes we have a database of 1000 customers, with
0 customers with income = low,
990 customers with income = medium,
10 customers with income = high.

The uncounted (uncorrected) probabilities are 0, 0.990, and 0.010. Applying the Laplacian correction — pretend we saw one extra customer at each income value, so the denominator becomes $1000 + 3 = 1003$:

$$\frac{1}{1003} = 0.001, \qquad \frac{991}{1003} = 0.988, \qquad \frac{11}{1003} = 0.011.$$

The corrected estimates are almost identical to the originals, but the troublesome zero has disappeared. With a large training set, adding one to each count changes the estimates only negligibly while conveniently avoiding zero probabilities.

8. Summary of the procedure

Estimate the **prior** $P(C_i)$ of each class from the fraction of training tuples in that class.

For the new tuple, estimate each **conditional probability** $P(x_k \mid C_i)$ by counting (categorical) or with a Gaussian (continuous).

Multiply them to get the **likelihood** $P(\mathbf{X} \mid C_i) = \prod_k P(x_k \mid C_i)$.

Compute $P(\mathbf{X} \mid C_i)\, P(C_i)$ for every class and **predict the class with the largest value.**

Use the **Laplacian correction** to prevent any zero-count probability from cancelling the product.

The method is simple, fast, and needs little training data; its main limitation is the independence assumption, which is rarely exactly true but works well in practice for many problems.