GAUSSIAN NAÏVE BAYES

Complete Tutorial — Theory, Fully Verified Numerical, and Interview-Depth Concepts

1. From Bayes' Theorem to "Naïve" Bayes

Bayes' theorem gives the posterior probability of a class given the observed features X:

$$P(\text{Class} \mid X) = \frac{P(X \mid \text{Class})\,P(\text{Class})}{P(X)} \;\propto\; P(X \mid \text{Class})\,P(\text{Class})$$

Since P(X) is the same for every class, it can be dropped when comparing classes — we only need the **numerator** to rank classes and pick the largest. The word **"naïve"** refers to one specific assumption: that all features are **conditionally independent** given the class, so their joint likelihood factorises into a simple product:

$$P(X \mid \text{Class}) = P(x_1, x_2, \ldots, x_n \mid \text{Class}) = \prod_{i=1}^{n} P(x_i \mid \text{Class})$$

Why this matters: this independence assumption is almost always false in real data (e.g. Age and Income are typically correlated) — yet Naïve Bayes remains highly effective in practice because classification only requires **ranking** P(Class|X) correctly, not estimating it accurately. A classifier can be badly calibrated (wrong probability values) but still make the correct decision every time, as long as the ranking of classes is preserved.

2. Why the Gaussian Distribution for Continuous Features?

For a categorical feature, $P(x_i|\text{Class})$ is just a frequency count. For a **continuous** feature (Age, Income), there is no direct frequency to count — so Gaussian Naïve Bayes **assumes** each feature is normally distributed within each class, and estimates the likelihood using the Gaussian probability density function (PDF):

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

where μ and σ are the mean and standard deviation of that feature, **estimated separately for each class** from the training data:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \qquad \text{(population / MLE variance, divide by } n\text{, not } n-1\text{)}$$

Classic interview trap: scikit-learn's GaussianNB (and this tutorial) divides by **n**, not n−1 — this is the **population** variance (equivalently, the Maximum Likelihood Estimate of variance), **not** the unbiased sample variance used in most introductory statistics courses. Getting this distinction right — and knowing scikit-learn's convention — is a common differentiator in interviews.

3. Worked Numerical Example

Step 1 — Training data

ID	Age	Income (₹k)	Buys_Computer
1	25	40	Yes
2	30	42	Yes
3	35	45	Yes
4	40	70	No
5	45	65	No
6	50	80	No

Step 2 — Tuple to classify

Let X = (Age = 38, Income = 50). We must decide: Yes or No?

Step 3 — Mean (μ) and standard deviation (σ) per class, per feature

For class "Yes" (tuples 1, 2, 3):

$$\mu_{\text{Yes,Age}} = \frac{25+30+35}{3} = 30$$

$$\sigma_{\text{Yes,Age}} = \sqrt{\frac{(25-30)^2+(30-30)^2+(35-30)^2}{3}} = \sqrt{16.667} \approx 4.0825$$

$$\mu_{\text{Yes,Income}} = \frac{40+42+45}{3} \approx 42.333$$

$$\sigma_{\text{Yes,Income}} = \sqrt{\frac{(40-42.33)^2+(42-42.33)^2+(45-42.33)^2}{3}} = \sqrt{4.222} \approx 2.0548$$

For class "No" (tuples 4, 5, 6):

$$\mu_{\text{No,Age}} = \frac{40+45+50}{3} = 45$$

$$\sigma_{\text{No,Age}} = \sqrt{\frac{(40-45)^2+(45-45)^2+(50-45)^2}{3}} = \sqrt{16.667} \approx 4.0825$$

$$\mu_{\text{No,Income}} = \frac{70+65+80}{3} \approx 71.667$$

$$\sigma_{\text{No,Income}} = \sqrt{\frac{(70-71.67)^2+(65-71.67)^2+(80-71.67)^2}{3}} = \sqrt{38.889} \approx 6.2361$$

Class	Feature	μ	σ
Yes	Age	30.00	4.0825
Yes	Income	42.33	2.0548
No	Age	45.00	4.0825
No	Income	71.67	6.2361

4. Step 4 — Gaussian Likelihoods

Class = Yes

$$P(\text{Age=38} \mid \text{Yes}) = \frac{1}{\sqrt{2\pi}\,(4.0825)}\, e^{-\frac{(38-30)^2}{2(4.0825)^2}} = \frac{1}{10.236}\, e^{-1.922} \approx 0.01433$$

$$P(\text{Income=50} \mid \text{Yes}) = \frac{1}{\sqrt{2\pi}\,(2.0548)}\, e^{-\frac{(50-42.33)^2}{2(2.0548)^2}} = \frac{1}{5.153}\, e^{-6.998} \approx 1.842 \times 10^{-4}$$

$$P(X \mid \text{Yes}) = 0.01433 \times 1.842 \times 10^{-4} \approx 2.638 \times 10^{-6}$$

Class = No

$$P(\text{Age=38} \mid \text{No}) = \frac{1}{\sqrt{2\pi}\,(4.0825)}\, e^{-\frac{(38-45)^2}{2(4.0825)^2}} = \frac{1}{10.236}\, e^{-1.472} \approx 0.02247$$

$$P(\text{Income=50} \mid \text{No}) = \frac{1}{\sqrt{2\pi}\,(6.2361)}\, e^{-\frac{(50-71.67)^2}{2(6.2361)^2}} = \frac{1}{15.635}\, e^{-6.033} \approx 1.530 \times 10^{-4}$$

$$P(X \mid \text{No}) = 0.02247 \times 1.530 \times 10^{-4} \approx 3.438 \times 10^{-6}$$

5. Priors and Posterior Probabilities

With 3 "Yes" and 3 "No" examples out of 6, the priors are P(Yes) = P(No) = 0.5. Multiplying by the likelihoods gives the **unnormalised** posteriors:

$$P(\text{Yes} \mid X) \propto P(X \mid \text{Yes})P(\text{Yes}) = 2.638 \times 10^{-6}(0.5) = 1.319 \times 10^{-6}$$

$$P(\text{No} \mid X) \propto P(X \mid \text{No})P(\text{No}) = 3.438 \times 10^{-6}(0.5) = 1.719 \times 10^{-6}$$

Since P(No|X) > P(Yes|X), the classifier already predicts **"No."** But a complete answer — and what interviewers often ask for next — is the **actual normalised probability**, obtained by dividing each unnormalised value by their sum:

$$P(\text{Yes} \mid X) = \frac{1.319 \times 10^{-6}}{1.319 \times 10^{-6} + 1.719 \times 10^{-6}} \approx 0.4342 \ (43.4\%)$$

$$P(\text{No} \mid X) = \frac{1.719 \times 10^{-6}}{1.319 \times 10^{-6} + 1.719 \times 10^{-6}} \approx 0.5658 \ (56.6\%)$$

Final Prediction: the classifier predicts **"No"** (does not buy a computer), with **56.6% confidence** — a genuinely close call, not an overwhelming one. This nuance (43% vs 57%, not 0% vs 100%) is exactly the kind of detail a strong candidate points out unprompted: the model is fairly uncertain here because X = (38, 50) sits roughly between the two class centres.

Independently verified against scikit-learn's GaussianNB() on this exact dataset: predict_proba returns [No: 0.5658, Yes: 0.4342] — an exact match.

6. Interesting Concepts

(a) The log-space trick (numerical stability)

Notice the likelihoods above are already tiny (~10^{-6}). With more features, multiplying many small probabilities together causes **numerical underflow** — the product silently becomes exactly 0 in floating-point arithmetic. The standard fix, used internally by every production Naïve Bayes implementation, is to work in **log-space**, where products become sums:

$$\ln P(\text{Class} \mid X) \ \propto \ \sum_{i=1}^{n} \ln P(x_i \mid \text{Class}) + \ln P(\text{Class})$$

Since log is monotonic, comparing log-posteriors gives exactly the same ranking (and hence the same decision) as comparing raw posteriors — but without ever risking underflow.

(b) The decision boundary — link to LDA/QDA

Setting $P(\text{Class}_0|X) = P(\text{Class}_1|X)$ for a single Gaussian feature and solving gives the boundary condition:

$$\frac{(x-\mu_0)^2}{2\sigma_0^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2} = \ln\frac{\sigma_1}{\sigma_0} + \ln\frac{P(\text{Class}_1)}{P(\text{Class}_0)}$$

- **If $\sigma_0 \neq \sigma_1$** (as in this example, 4.08 vs 6.24 for Age, 2.05 vs 6.24 for Income), the x^2 terms do **not** cancel — the boundary is **quadratic**. This is exactly **Quadratic Discriminant Analysis (QDA)**.
- If a **shared** σ is assumed across classes, the x^2 terms cancel and the boundary becomes **linear** — this is **Linear Discriminant Analysis (LDA)**, and (as derived in the Logistic Regression interview set) it is also exactly the point where Gaussian Naïve Bayes and Logistic Regression become the same model.

(c) Zero-probability / zero-variance handling

If a feature has exactly zero variance within a class (or a categorical feature's value never appears in the training data for that class), the Gaussian PDF or the frequency count becomes exactly 0 — multiplying it into the product forces the **entire** posterior to 0, regardless of how strongly the other features favour that class. Scikit-learn's GaussianNB fixes this with **var_smoothing**: a tiny value added to every variance:

$$\sigma^2 \ \leftarrow \ \sigma^2 + \epsilon, \qquad \epsilon = \text{var\_smoothing} \times \max(\text{feature variance})$$

For categorical/multinomial Naïve Bayes, the analogous fix is **Laplace (add-one) smoothing** on the frequency counts — the same underlying problem, solved the same way: never let a single unseen or zero-variance feature veto an entire class.

(d) Generative vs. Discriminative

Naïve Bayes is a **generative** classifier — it models P(X|Class) and P(Class) and combines them via Bayes' theorem to get P(Class|X). Logistic regression is **discriminative** — it models P(Class|X) directly, with no assumption about how X itself is distributed. Being generative means Naïve Bayes can also **generate synthetic data** by sampling from its fitted P(X|Class), something a discriminative model cannot do.

7. Summary

- Naïve Bayes ranks classes using Bayes' theorem under a conditional-independence assumption across features.
- For continuous features, likelihoods are estimated using the Gaussian PDF, with μ and σ computed **per class** (population variance, divide by n).
- Always report the **normalised** posterior probability, not just the winning class — it reveals model confidence.

- Production systems compute in log-space to avoid underflow; zero-variance/zero-frequency features are handled by smoothing.
- With a shared variance across classes, Gaussian Naïve Bayes and Logistic Regression become the same linear model — the classic generative/discriminative pairing.