IN402

Machine Learning

Chapter

7

Naive Bayes Classifiers & Bayesian Networks

Author: Abbas El-Hajj Youssef

University: Lebanese University

Department: Department of Computer Science

These notes extend course materials taught by Prof. Ahmad Faour with additional content from textbooks and supplementary resources.

Disclaimer: *This is not an official course document.*

Contents

1 Introduction: Making Decisions Under Uncertainty

It's 3 AM in the emergency department. A patient arrives with respiratory symptoms: fever at 38.9°C, productive cough, and abnormal crackling sounds in the lungs. The physician faces a decision.

Is this bacterial pneumonia requiring immediate antibiotics? Or a viral infection where antibiotics would be useless—even harmful?

No single symptom provides certainty. High fever suggests bacterial infection, but viral patients sometimes run high fevers too. Productive cough points toward bacterial, but isn't definitive. Lung crackles are concerning, but appear in both conditions.

Each symptom is evidence. None is conclusive. The physician must *combine* these uncertain pieces into a decision.

This is **probabilistic classification**: systematically combining imperfect evidence to make the best possible choice. And it's exactly what we'll learn to do mathematically.

1.1 Learning from Experience

How does an experienced physician decide? Through pattern recognition built from hundreds of past cases. They've seen bacterial pneumonia present with high fever and productive cough. They've seen viral infections with milder symptoms. Over time, they've internalized the patterns.

We can formalize this. Suppose the hospital has records of 200 previous respiratory cases, each with a confirmed diagnosis:

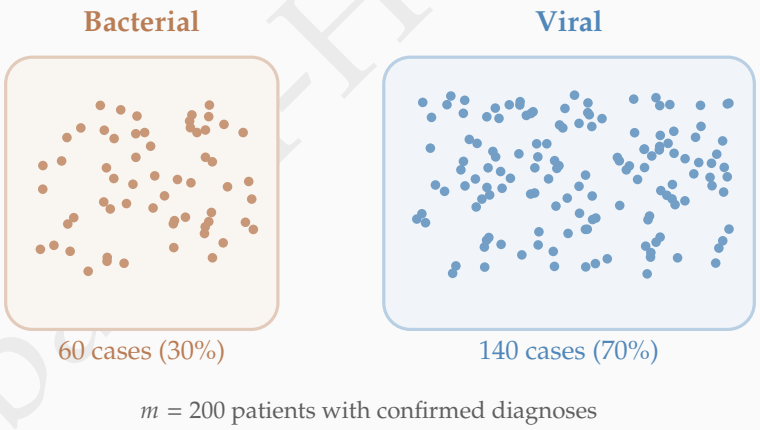


Figure 1: *Historical data: 200 past cases form the basis for learning diagnostic patterns.*

For each patient, three symptoms were recorded:

- **Fever**: High (above 39°C) or Normal
- **Cough**: Productive (with mucus) or Dry
- **Lung sounds**: Crackles present or Clear

How do we use these 200 past cases to diagnose a *new* patient?

1.2 A First Attempt: Pattern Memorization

The most straightforward approach: find patients in our database with the exact same symptoms and see what diagnosis they had.

With three binary symptoms, there are $2^3 = 8$ possible combinations:

Fever	Cough	Sounds	Cases	Likely Diagnosis
High	Prod.	Crackles	35	Bacterial
High	Prod.	Clear	8	Mixed
High	Dry	Crackles	12	Viral
High	Dry	Clear	18	Viral
Norm.	Prod.	Crackles	4	Bacterial
Norm.	Prod.	Clear	45	Viral
Norm.	Dry	Crackles	3	???
Norm.	Dry	Clear	75	Viral

Only 3 cases!
Too few to trust.

Figure 2: *Counting each symptom combination. Some have abundant data; others are sparse.*

This works for common combinations. But notice the highlighted row: only 3 patients had (Normal fever, Dry cough, Crackles). Too few for reliable conclusions.
And this is with just 3 symptoms. What happens as we add more?

1.3 The Curse of Dimensionality

Real diagnostic systems track many more symptoms. A thorough workup might include:

- Fever level, cough type, lung sounds
- Chest pain, shortness of breath, fatigue
- White blood cell count, oxygen saturation
- Patient age group, smoking history, recent travel

With 10 binary features, there are $2^{10} = 1,024$ possible combinations. With 20 features: over one million.

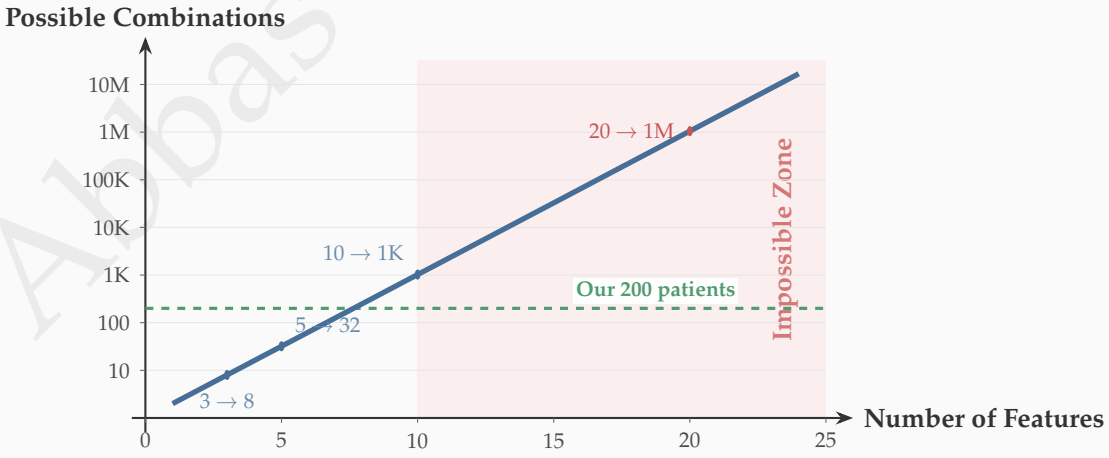


Figure 3: *Combinations grow exponentially; data cannot keep up. Beyond a handful of features, most combinations never appear in training.*

The math is brutal. With 200 patients and 10 features:

$$\frac{200 \text{ patients}}{1,024 \text{ combinations}} \approx 0.2 \text{ patients per combination}$$

Most combinations have *zero* observations. When a new patient arrives with an unseen combination, memorization has nothing to say.
This isn't a "collect more data" problem. Even with 10,000 patients and 20 features:

$$\frac{10,000}{1,048,576} \approx 0.01 \text{ patients per combination}$$

Data grows linearly; combinations grow exponentially. Exponential always wins.

1.4 A Different Strategy

We need to abandon memorization. Instead of asking "have we seen this exact pattern before?", we ask:
How does each symptom individually relate to each diagnosis?
This changes everything. Instead of tracking a million combinations, we track individual relationships:

- How common is high fever among bacterial patients? Among viral patients?
- How common is productive cough in each group?
- And so on for each symptom.

Then, when a new patient arrives, we *combine* these individual pieces using the rules of probability.

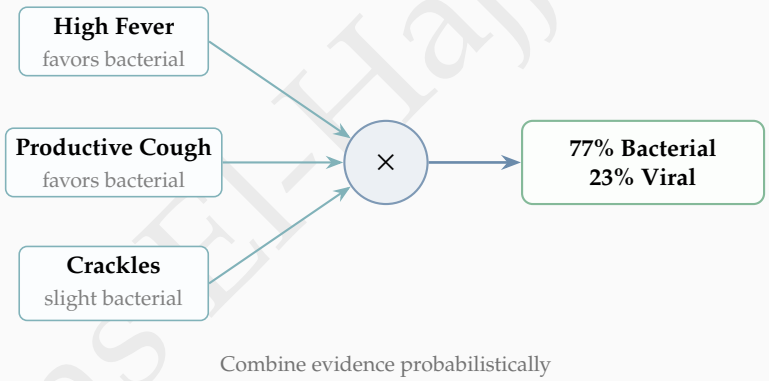


Figure 4: *The Naive Bayes strategy: assess each symptom independently, then combine.*

This approach can make predictions for *any* symptom combination, even ones never seen in training. It generalizes from individual symptom patterns to complete diagnoses.
The key insight is treating each symptom as independent evidence. This assumption—called **conditional independence**—is often technically wrong. Symptoms can correlate for reasons beyond the diagnosis. But the assumption makes the math tractable, and surprisingly, it works well in practice.
This is **Naive Bayes**: "naive" because of the independence assumption, "Bayes" because of the probabilistic framework for combining evidence.

1.5 Roadmap

We'll build this framework step by step:

1. **One symptom**: Learn Bayesian reasoning with a single piece of evidence. (Section 2)

2. **Multiple symptoms**: Scale up using independence. See how evidence combines. (Section 3)
3. **Complete classifier**: Assemble the full algorithm with examples. (Section 4)
4. **Zero probabilities**: Handle missing data through smoothing. (Section 5)
5. **Continuous measurements**: Extend to real-valued features. (Section 6)
6. **Numerical stability**: Prevent computational underflow using logarithms. (Section 7)
7. **Why it works**: Understand why Naive Bayes succeeds despite wrong assumptions. (Section 8)
8. **Beyond independence**: Explore Bayesian Networks for modeling dependencies. (Section 9)
9. **Synthesis**: Bring together the complete framework with practical guidance. (Section 10)

By the end, you'll have a complete probabilistic classification framework—one that physicians, spam filters, and countless other systems use daily—plus deep insight into why it works and how to extend it when needed.

2 Foundation: Reasoning with One Symptom

Before tackling multiple symptoms, let's master the logic with just one: fever status. The concepts here—prior, likelihood, posterior—carry through everything that follows.

2.1 What We Know Before Examining the Patient

Before any symptoms are observed, what can we say? We can look at base rates.
Of our 200 historical cases:

- 60 were bacterial pneumonia (30%)
- 140 were viral infections (70%)

If we had to guess a diagnosis *before* seeing any symptoms, we'd predict viral—it's more than twice as common. We'd be right 70% of the time.
This baseline is called the **prior probability**—our belief before observing evidence:

$$P(\text{Bacterial}) = 0.30 \qquad P(\text{Viral}) = 0.70$$

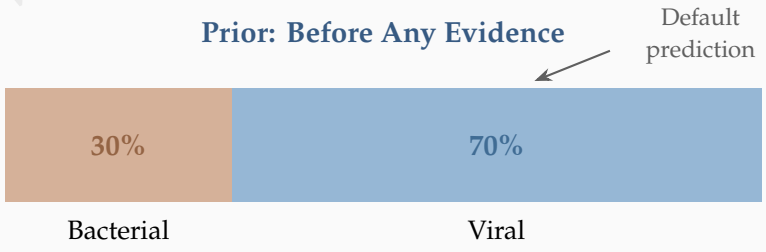


Figure 5: *The prior distribution. Without symptoms, predict viral—you'll be right 70% of the time.*

The prior is just our starting point. The real power comes from updating these beliefs with symptoms.

2.2 How Fever Behaves Within Each Diagnosis

Now suppose we observe: **the patient has a high fever**.
Intuitively, this should shift our belief toward bacterial pneumonia—bacterial infections tend to cause higher fevers. But by how much?
To quantify this, we examine our historical data:

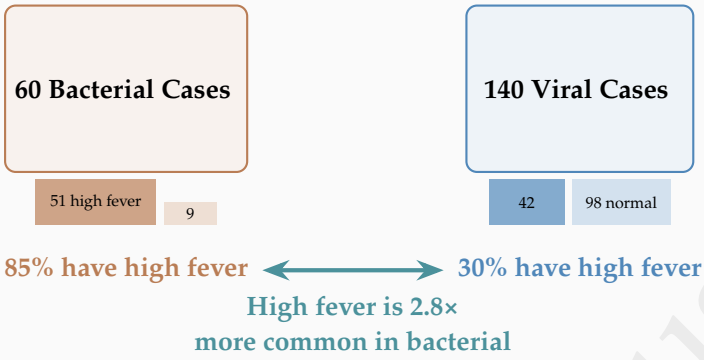


Figure 6: *Fever patterns differ dramatically between diagnoses. This difference makes fever diagnostically useful.*

We've discovered something important:

$$P(\text{High Fever} \mid \text{Bacterial}) = \frac{51}{60} = 0.85$$

$$P(\text{High Fever} \mid \text{Viral}) = \frac{42}{140} = 0.30$$

These are **likelihoods**—they tell us how probable the observed symptom is under each diagnosis. The vertical bar "|" means "given" or "conditional on."
The likelihood ratio is striking: $0.85/0.30 = 2.83$. High fever is nearly three times more common in bacterial cases. This is strong diagnostic evidence.

Likelihood Is Not What We Want (Yet)

A common confusion: the likelihood $P(\text{High Fever} \mid \text{Bacterial}) = 0.85$ does *not* mean there's an 85% chance the patient has bacterial pneumonia.
The likelihood asks: "Among patients *we know* have bacterial pneumonia, what fraction have high fever?" That's a question about fever, not diagnosis.
What we want is the reverse: "Among patients with high fever, what fraction have bacterial pneumonia?" That's the **posterior probability**—computing it requires one more step.

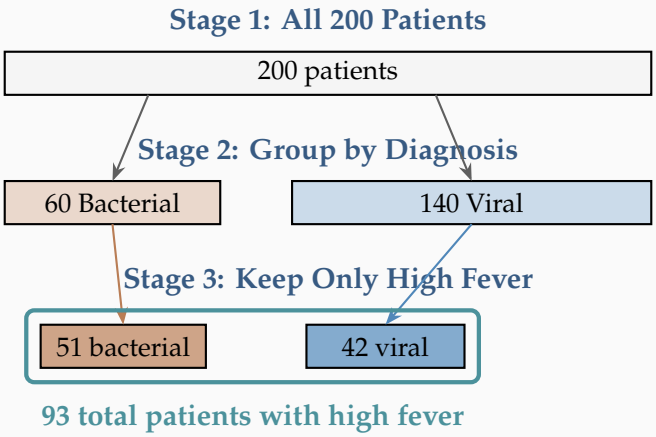
2.3 Combining Prior and Likelihood

We now have two pieces of information:

- **Prior**: Bacterial is less common overall (30% vs 70%)
- **Likelihood**: High fever is much more common in bacterial cases (85% vs 30%)

These pull in opposite directions. The prior favors viral; the evidence favors bacterial. How do we reconcile them?

Think through this carefully. Imagine the 200 patients laid out before us:



$$P(\text{Bacterial} \mid \text{High Fever}) = \frac{51}{93} = 54.8\%$$

Figure 7: *Tracing through the data: among patients with high fever, 54.8% have bacterial pneumonia.*

The logic:

1. Start with all 200 patients
2. Bacterial cases: 60, of which 51 have high fever (85%)
3. Viral cases: 140, of which 42 have high fever (30%)
4. Total with high fever: $51 + 42 = 93$
5. Fraction that are bacterial: $51/93 = 54.8\%$

Observing high fever shifted our belief from 30% (prior) to 55% (posterior). The evidence wasn't strong enough to make bacterial overwhelmingly likely, but it flipped our prediction from viral to bacterial.

2.4 Bayes' Theorem: The General Formula

Let's express that counting argument algebraically. The number of bacterial patients with high fever is:

$$60 \times 0.85 = P(\text{Bacterial}) \times P(\text{High Fever} \mid \text{Bacterial}) \times 200$$

So our posterior is:

$$P(\text{Bacterial} \mid \text{High Fever}) = \frac{P(\text{Bact}) \times P(\text{High} \mid \text{Bact})}{P(\text{Bact}) \times P(\text{High} \mid \text{Bact}) + P(\text{Viral}) \times P(\text{High} \mid \text{Viral})}$$

The denominator is just the total probability of observing high fever, written as $P(\text{High Fever})$. This is **Bayes' theorem**:

$$\boxed{P(\text{Class} \mid \text{Evidence}) = \frac{P(\text{Evidence} \mid \text{Class}) \times P(\text{Class})}{P(\text{Evidence})}}$$

In words: the posterior is proportional to the likelihood times the prior.

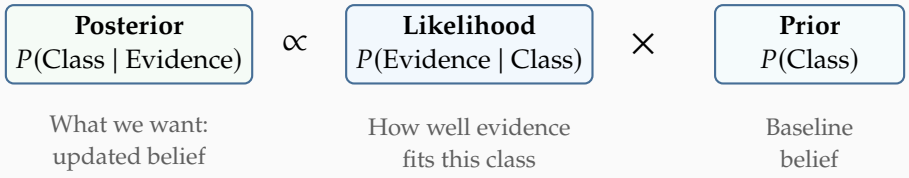


Figure 8: *Bayes' theorem decomposes the posterior into likelihood (how diagnostic is the evidence?) and prior (how common is the class?).*

2.5 The Update in Pictures

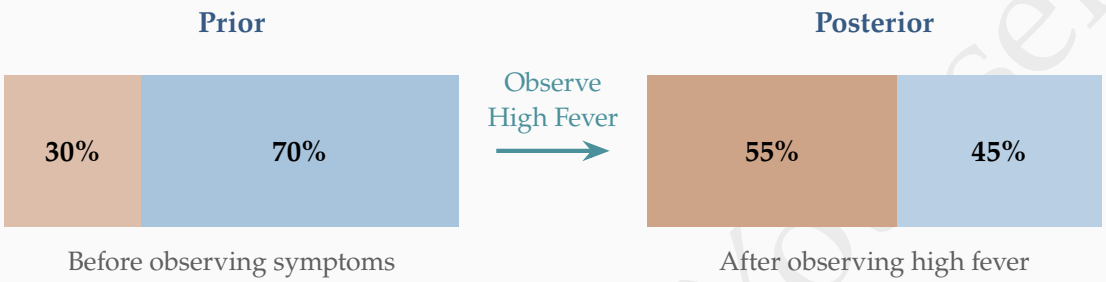


Figure 9: *High fever shifts the probability from 30% to 55%. The diagnosis flips from viral to bacterial.*

The evidence was substantial but not overwhelming. A single symptom shifted belief by 25 percentage points—enough to change the diagnosis, but leaving meaningful uncertainty.

2.6 Making a Decision

With probabilities in hand, classification is simple: predict the class with higher posterior probability.

$$P(\text{Bacterial} \mid \text{High Fever}) = 0.548$$
$$P(\text{Viral} \mid \text{High Fever}) = 0.452$$

Since $0.548 > 0.452$, we predict **Bacterial**.
This single-feature classifier captures something real. High fever genuinely is more common in bacterial infections, and our method quantifies exactly how much that evidence should shift our beliefs.
But one symptom isn't enough for confident diagnosis. Our posterior is only 55%—barely better than a coin flip. We need more evidence.

The Bayesian Update

Three quantities determine the posterior:

1. **Prior** $P(\text{Class})$: How common is this class overall?
2. **Likelihood** $P(\text{Evidence} \mid \text{Class})$: How typical is this evidence for this class?
3. **Posterior** $P(\text{Class} \mid \text{Evidence})$: Our updated belief after seeing evidence

The update formula:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalizing Constant}}$$

Strong evidence (high likelihood ratio) can overcome an unfavorable prior. Weak evidence barely shifts beliefs. This is rational belief revision—exactly how we'd want a diagnostic system to reason.

3 Scaling to Multiple Symptoms

A real patient doesn't present with just one symptom. Our 3 AM patient has high fever *and* productive cough *and* lung crackles. Each symptom is evidence, and we need to combine them all.

3.1 The Goal

We want to compute:

$$P(\text{Bacterial} \mid \text{High Fever, Productive Cough, Crackles})$$

By Bayes' theorem:

$$P(\text{Bacterial} \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid \text{Bacterial}) \times P(\text{Bacterial})}{P(\mathbf{x})}$$

where $\mathbf{x}$ denotes the complete symptom pattern (High, Productive, Crackles).
The prior $P(\text{Bacterial}) = 0.30$ is easy. The challenge is the likelihood $P(\mathbf{x} \mid \text{Bacterial})$—the probability of observing this *entire* symptom combination among bacterial patients.

3.2 Why Direct Counting Fails

We could estimate this directly: look through the 60 bacterial cases and count how many have exactly (High Fever, Productive Cough, Crackles).
Suppose we find 35 such cases. Then:

$$P(\text{High, Prod, Crackles} \mid \text{Bacterial}) = \frac{35}{60} = 0.583$$

This seems reasonable. But try another combination.
How many bacterial cases have (Normal Fever, Dry Cough, Crackles)?
Looking through the data... zero. Not a single bacterial patient with this combination.

$$P(\text{Normal, Dry, Crackles} \mid \text{Bacterial}) = \frac{0}{60} = 0$$

We just estimated a probability as exactly zero. This creates a catastrophic problem.

The Zero Probability Disaster

A zero probability has devastating consequences. If a future patient has this symptom combination:

$$P(\text{Bacterial} \mid \mathbf{x}) \propto P(\mathbf{x} \mid \text{Bacterial}) \times P(\text{Bacterial})$$
$$= 0 \times 0.30 = 0$$

No matter how strong the other evidence, bacterial pneumonia can *never* be diagnosed for any patient with this symptom pattern. The classifier is permanently blind.
But absence of evidence is not evidence of absence. We didn't observe this combination in 60 cases—that doesn't mean it's medically impossible.

This isn't an edge case—it's the norm with limited data:

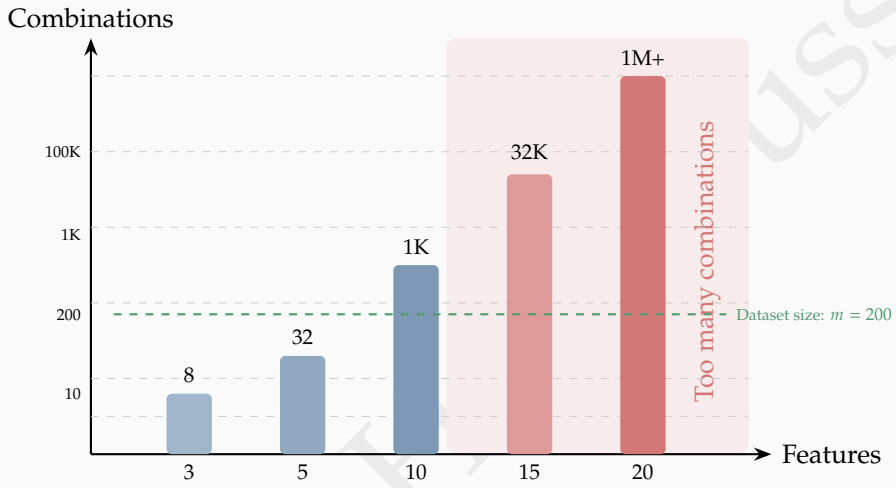


Figure 10: *Combinatorial explosion: possible feature combinations quickly exceed what data can cover.*

Even with just 10 binary features:

- 1,024 possible combinations
- 200 patients spread across them
- Average: 0.2 patients per combination
- Reality: a few common patterns get most patients; the rest get zero

Direct counting fundamentally cannot work for realistic numbers of features.

3.3 The Independence Assumption: Breaking the Problem Apart

Here's the breakthrough idea. Instead of estimating the joint probability of all symptoms together:

1. Estimate how each symptom behaves *individually* within each diagnosis
2. Assume symptoms provide *independent* evidence
3. Multiply the individual probabilities together

Under this assumption:

$$P(\text{Fever, Cough, Sounds} \mid \text{Class}) = P(\text{Fever} \mid \text{Class}) \times P(\text{Cough} \mid \text{Class}) \times P(\text{Sounds} \mid \text{Class})$$

Now instead of estimating one quantity from (potentially zero) exact matches, we estimate three quantities, each from much larger data pools.

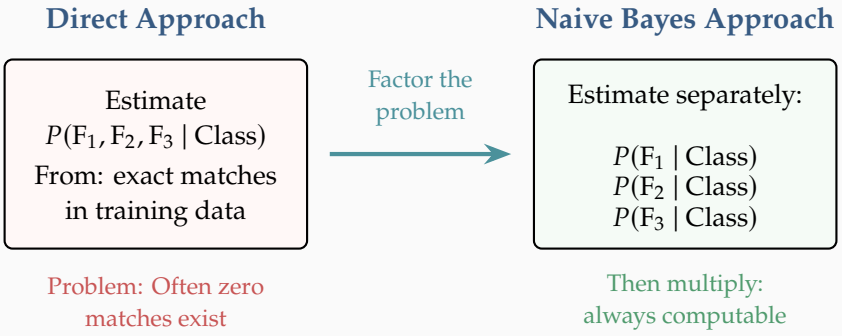


Figure 11: *The Naive Bayes insight: estimate simple quantities and combine them, rather than estimate complex quantities directly.*

3.4 What Does Independence Mean?

The assumption has a precise meaning: **conditional independence given the class**.
In our medical context: once we know the diagnosis (bacterial or viral), learning one symptom tells us nothing additional about the others.
Example: Suppose we know a patient has bacterial pneumonia. Does learning they have high fever change our belief about their cough type?
Under conditional independence: *No*. The diagnosis "explains" both symptoms. Fever and cough are both consequences of the bacterial infection, but they don't directly influence each other.

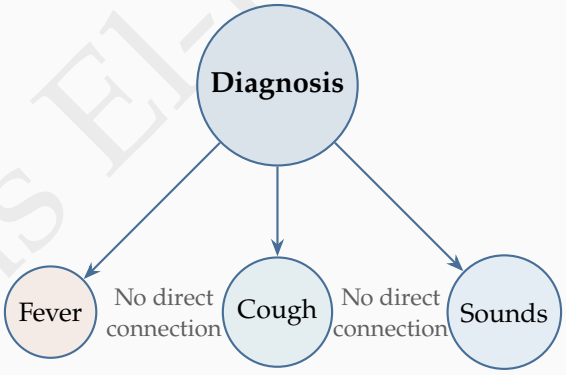


Figure 12: *The Naive Bayes assumption: the diagnosis causes each symptom, but symptoms don't directly affect each other.*

Conditional Independence

Features $X_1, \ldots, X_n$ are **conditionally independent given class** Y if:

$$P(X_1, \ldots, X_n \mid Y = c) = \prod_{j=1}^{n} P(X_j \mid Y = c)$$

Meaning: Once you know the diagnosis (Y), learning one symptom tells you nothing additional about the others.

Equivalently: For any two features X_i, X_j:

$$P(X_i \mid X_j, Y = c) = P(X_i \mid Y = c)$$

Knowing X_j doesn't change our belief about X_i once we know the class.

Manufacturing Analogy

Common confusion: "Independence means features are unrelated"
NO! It means *conditionally* independent given the class.
Imagine a factory producing light bulbs. Two tests:

- X_1: Brightness test
- X_2: Lifespan test

Unconditionally: X_1 and X_2 are correlated!

- Bright bulbs tend to have longer lifespans
- Dim bulbs tend to fail quickly

But conditionally:
Given bulb is Defective:

- Knowing brightness is low doesn't tell us anything extra about lifespan
- Both are independently affected by the defect

Given bulb is Good:

- Both pass independently

The common cause (defect/good) explains the correlation. Once we know the cause, features become independent.

3.5 Is This Assumption True?

Honestly? Usually not.
In medicine, symptoms *do* correlate beyond what the diagnosis explains:

- Fever and chills go together (temperature regulation)
- Productive cough and lung sounds are mechanistically linked
- Fatigue correlates with almost everything

So why use this clearly false assumption?
Because it works anyway.
The assumption is wrong, but the classifier often makes correct predictions despite that. We'll understand why in Section 8. For now, accept that this "naive" assumption trades some accuracy in probability estimates for dramatic gains in reliability and data efficiency.

3.6 The Computational Payoff

The independence assumption transforms an intractable problem into an easy one.

Without independence (direct estimation):

- Parameters needed: one probability for each possible symptom combination
- With n binary features: 2^n parameters per class
- With 20 features: over 1 million parameters per class

With independence (Naive Bayes):

- Parameters needed: one probability for each feature value, per class
- With n binary features: n parameters per class
- With 20 features: just 20 parameters per class

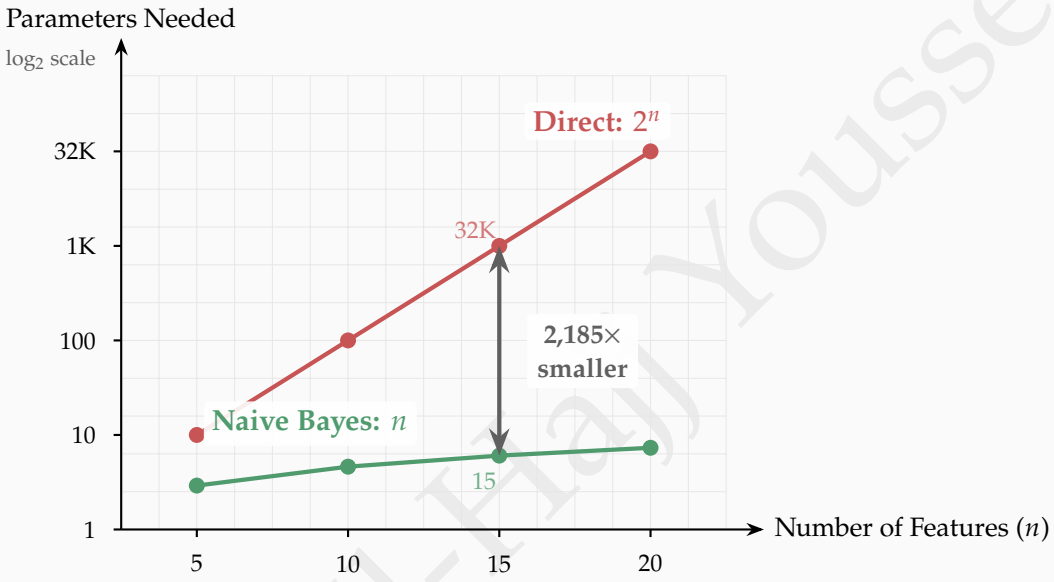


Figure 13: *Parameter complexity: Direct estimation grows exponentially (2^n) while Naive Bayes grows linearly (n). With 15 features, Naive Bayes needs 2,185× fewer parameters.*

More parameters means more data needed for reliable estimation. By slashing the parameter count, Naive Bayes works with far less data than methods modeling all feature interactions.

4 The Complete Naive Bayes Classifier

We now have all the conceptual pieces. Time to assemble them into a complete, working classifier.

4.1 The Classification Rule

Given a patient with symptoms $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we want the most probable diagnosis:

$$\hat{y} = \arg\max_c P(Y = c \mid \mathbf{x})$$

By Bayes' theorem:

$$P(Y = c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid Y = c) \cdot P(Y = c)}{P(\mathbf{x})}$$

The denominator $P(\mathbf{x})$ is the same for all classes—it just ensures probabilities sum to one. For finding the maximum, we can ignore it:

$$P(Y = c \mid \mathbf{x}) \propto P(\mathbf{x} \mid Y = c) \cdot P(Y = c)$$

Apply the independence assumption to factor the likelihood:

$$P(\mathbf{x} \mid Y = c) = P(x_1 \mid Y = c) \times P(x_2 \mid Y = c) \times \cdots \times P(x_n \mid Y = c)$$

Putting it together:

$$\boxed{\hat{y} = \arg\max_c \left[P(Y = c) \prod_{j=1}^{n} P(x_j \mid Y = c) \right]}$$

This is the Naive Bayes classification rule. Compute a score for each class by multiplying the prior by all likelihoods, then pick the class with the highest score.

4.2 What We Need to Learn

The classifier requires two types of parameters, both estimated from training data:
1. Prior probabilities — How common is each class?

$$P(Y = c) = \frac{\text{number of training examples with class } c}{\text{total training examples}}$$

2. Likelihoods — How does each feature behave within each class?

$$P(X_j = v \mid Y = c) = \frac{\text{number of class-}c\text{ examples where feature } j \text{ has value } v}{\text{number of class-}c\text{ examples}}$$

That's it. Count occurrences and divide. No optimization, no iteration, no gradients—just counting.

4.3 A Complete Worked Example

Let's classify our 3 AM patient step by step.
The patient: High fever, Productive cough, Lung crackles
Training data summary (from our 200 cases):

Feature	Bacterial (60)	Viral (140)
Fever = High	51/60 = **0.850**	42/140 = **0.300**
Cough = Productive	52/60 = **0.867**	67/140 = **0.479**
Sounds = Crackles	48/60 = **0.800**	73/140 = **0.521**
Prior	60/200 = **0.300**	140/200 = **0.700**

Figure 14: *All parameters needed: priors and likelihoods for each feature-class combination.*

Step 1: Compute the bacterial score

$$\text{Score(Bacterial)} = P(\text{Bact}) \times P(\text{High} \mid \text{Bact}) \times P(\text{Prod} \mid \text{Bact}) \times P(\text{Crackles} \mid \text{Bact})$$

$$= 0.300 \times 0.850 \times 0.867 \times 0.800 = 0.177$$

Step 2: Compute the viral score

$$\text{Score}(\text{Viral}) = P(\text{Viral}) \times P(\text{High} \mid \text{Viral}) \times P(\text{Prod} \mid \text{Viral}) \times P(\text{Crackles} \mid \text{Viral})$$
$$= 0.700 \times 0.300 \times 0.479 \times 0.521 = 0.052$$

Step 3: Compare and decide
Bacterial score (0.177) is higher than viral score (0.052).
Prediction: Bacterial pneumonia

4.4 Understanding the Evidence

Let's dissect how each piece of evidence contributed:

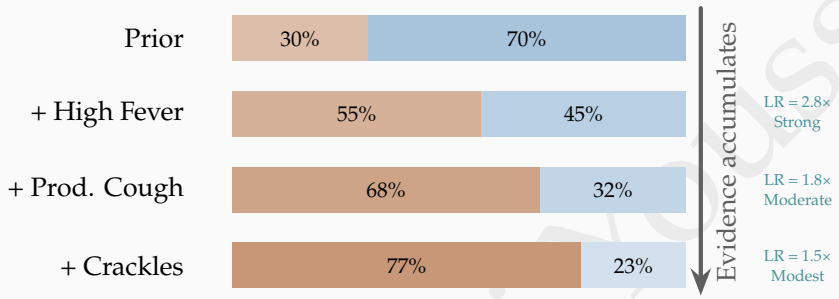


Figure 15: *Each symptom shifts the probability. Strong evidence (high likelihood ratio) shifts more; weak evidence shifts less.*

The likelihood ratio for each symptom measures its diagnostic value:

- **High fever**: 0.850/0.300 = 2.83 — Strong evidence for bacterial
- **Productive cough**: 0.867/0.479 = 1.81 — Moderate evidence
- **Crackles**: 0.800/0.521 = 1.54 — Modest evidence

Combined effect: $2.83 \times 1.81 \times 1.54 = 7.9$. Three pieces of moderate-to-strong evidence combine into very strong evidence.

4.5 Converting Scores to Probabilities

The raw scores (0.177 and 0.052) aren't probabilities—they don't sum to one. If we need actual posterior probabilities, we normalize:

$$P(\text{Bacterial} \mid \mathbf{x}) = \frac{0.177}{0.177 + 0.052} = \frac{0.177}{0.229} = 0.773 = 77.3\%$$

$$P(\text{Viral} \mid \mathbf{x}) = \frac{0.052}{0.229} = 0.227 = 22.7\%$$

The patient has 77% probability of bacterial pneumonia. That's strong enough to act on, though not certain—acknowledging remaining uncertainty is part of what makes probabilistic reasoning valuable.

4.6 Handling New Combinations

Here's where Naive Bayes shows its power. Suppose another patient presents with:

- Normal fever

- Productive cough
- Crackles

Did we see this exact combination in training? Maybe not. But we can still classify:

$$\begin{aligned}\text{Score}(\text{Bacterial}) &= 0.300 \times (1 - 0.850) \times 0.867 \times 0.800 \\ &= 0.300 \times 0.150 \times 0.867 \times 0.800 = 0.031\end{aligned}$$

$$\begin{aligned}\text{Score}(\text{Viral}) &= 0.700 \times (1 - 0.300) \times 0.479 \times 0.521 \\ &= 0.700 \times 0.700 \times 0.479 \times 0.521 = 0.122\end{aligned}$$

Viral wins ($0.122 > 0.031$). The normal fever strongly shifted evidence toward viral, outweighing the other symptoms.
The classifier generalized from individual symptom patterns to a combination it may have never seen. This is the payoff of the independence assumption.

4.7 A Second Domain: Email Spam Classification

Having mastered medical diagnosis, let's verify the framework generalizes. The mathematics should work identically in any domain.

Spam Detection with Word Features

A spam filter has learned these patterns from historical emails:
Priors:

- $P(\text{Spam}) = 0.3$ — 30% of emails are spam
- $P(\text{Legitimate}) = 0.7$ — 70% are legitimate

Likelihoods:

$$P(\text{“free"} \mid \text{Spam}) = 0.8 \qquad P(\text{“free"} \mid \text{Legitimate}) = 0.1$$
$$P(\text{“click here"} \mid \text{Spam}) = 0.6 \qquad P(\text{“click here"} \mid \text{Legitimate}) = 0.05$$

Task: A new email contains both “free" and “click here". Spam or legitimate?

Spam score:

$$\text{Score}(\text{Spam}) = 0.3 \times 0.8 \times 0.6 = 0.144$$

Legitimate score:

$$\text{Score}(\text{Legitimate}) = 0.7 \times 0.1 \times 0.05 = 0.0035$$

Since $0.144 > 0.0035$, predict **Spam**.
The likelihood ratio: spam is $0.144/0.0035 = 41$ times more likely given this evidence. Even though spam is less common overall (30% prior), the word combination is so diagnostic that posterior confidence reaches:

$$P(\text{Spam} \mid \text{email}) = \frac{0.144}{0.144 + 0.0035} \approx 0.976 = 97.6\%$$

The framework carries over perfectly. Whether symptoms or words, discrete or continuous, the logic remains: assess how each feature behaves within each class, then combine evidence multiplicatively.

The Naive Bayes Algorithm

Training (one pass through data):

1. Count class frequencies $\rightarrow$ priors $P(Y = c)$
2. For each feature and class, count value frequencies $\rightarrow$ likelihoods $P(X_j = v \mid Y = c)$

Prediction (for new instance $\mathbf{x}$):

1. For each class c: compute $\text{Score}(c) = P(Y = c) \prod_j P(x_j \mid Y = c)$
2. Predict: $\hat{y} = \arg\max_c \text{Score}(c)$
3. (Optional) Normalize scores to get probabilities

Complexity: Training is $O(mn)$ where m is training examples and n is features. Prediction is $O(nK)$ where K is the number of classes. Both are linear—no iterative optimization needed.
See Section 7 for handling numerical stability with many features.

5 Handling Zero Probabilities: Laplace Smoothing

Our classifier is elegant and efficient. But it has a critical vulnerability—one that can render it completely useless for certain inputs.

5.1 The Failure Mode

Suppose we include a fourth symptom: **wheezing**.
Looking through our training data:

- Bacterial cases with wheezing: 0 out of 60
- Viral cases with wheezing: 15 out of 140

Our likelihood estimates become:

$$P(\text{Wheezing} \mid \text{Bacterial}) = \frac{0}{60} = 0$$

$$P(\text{Wheezing} \mid \text{Viral}) = \frac{15}{140} = 0.107$$

Now a patient arrives with wheezing. What happens to the bacterial score?

$$\text{Score}(\text{Bacterial}) = P(\text{Bact}) \times P(\text{High} \mid \text{Bact}) \times P(\text{Prod} \mid \text{Bact}) \times P(\text{Crackles} \mid \text{Bact}) \times P(\text{Wheeze} \mid \text{Bact})$$

$$= 0.300 \times 0.850 \times 0.867 \times 0.800 \times \mathbf{0} = \mathbf{0}$$

Zero. No matter how strong the other evidence, the product is zero.

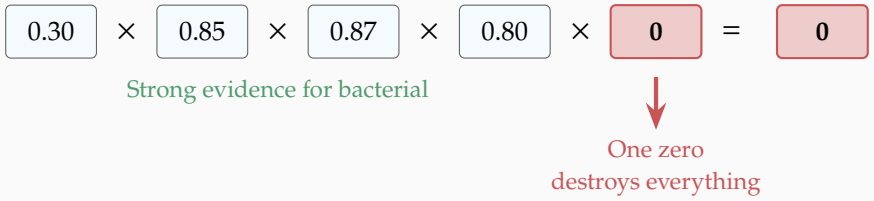


Figure 16: *A single zero probability nullifies all other evidence. The entire product collapses.*

This patient could have textbook bacterial symptoms—high fever, productive cough, crackles—but because they also wheeze, and we never saw wheezing in our 60 bacterial training cases, bacterial pneumonia becomes *impossible* to predict.

5.2 The Solution: Add Pseudocounts

The fix is elegant: pretend we saw every possible outcome at least a little bit.
Before any real data, imagine we observed α "fake" cases of each possible feature value in each class. This ensures no probability is ever exactly zero.
Without smoothing:

$$P(X = v \mid Y = c) = \frac{\text{count}(X = v, Y = c)}{\text{count}(Y = c)}$$

With smoothing (adding pseudocount α):

$$P(X = v \mid Y = c) = \frac{\text{count}(X = v, Y = c) + \alpha}{\text{count}(Y = c) + \alpha \cdot d}$$

where d is the number of possible values for feature X (for binary features, $d = 2$).
The denominator adjustment ($\alpha \cdot d$) ensures probabilities still sum to one.

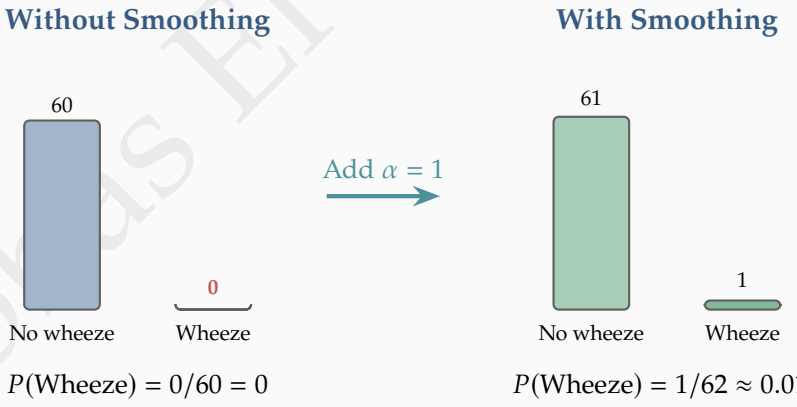


Figure 17: *Smoothing adds a small count to every outcome. Zero becomes small-but-nonzero.*

5.3 The Effect on Estimates

Smoothing changes our probability estimates. How much?
For rare/unseen values (where smoothing matters most):

- Without smoothing: $0/60 = 0$ — impossible
- With smoothing ($\alpha = 1$): $1/62 = 0.016$ — unlikely but possible

For common values (where we have abundant data):

- Without smoothing: $51/60 = 0.850$
- With smoothing: $52/62 = 0.839$

The change is tiny—less than 2%. When we have real data, it dominates. Pseudocounts only matter when real data is sparse.

Laplace Smoothing

Without smoothing (Maximum Likelihood):

$$P(X = v \mid Y = c) = \frac{\text{count}(X = v, Y = c)}{\text{count}(Y = c)}$$

With Laplace smoothing ($\alpha = 1$):

$$\boxed{P(X = v \mid Y = c) = \frac{\text{count}(X = v, Y = c) + 1}{\text{count}(Y = c) + d}}$$

where d = number of possible values for feature X.
Why $+d$ in denominator? To ensure probabilities sum to 1:

$$\sum_v P(X = v \mid Y = c) = \frac{\sum_v (\text{count}(v) + 1)}{\text{count}(Y = c) + d} = \frac{\text{count}(Y = c) + d}{\text{count}(Y = c) + d} = 1$$

5.4 Smoothing the Priors

The same logic applies to class priors:

$$P(Y = c) = \frac{\text{count}(Y = c) + \alpha}{m + \alpha \cdot K}$$

where K is the number of classes and m is total training examples.
For our example with $\alpha = 1$:

$$P(\text{Bacterial}) = \frac{60 + 1}{200 + 2} = \frac{61}{202} = 0.302$$

$$P(\text{Viral}) = \frac{140 + 1}{200 + 2} = \frac{141}{202} = 0.698$$

Minimal change from unsmoothed values (0.300 and 0.700), but now we're protected against edge cases.

The Bayesian Interpretation

Smoothing isn't ad-hoc—it has a principled interpretation.
Adding pseudocounts is equivalent to having a **prior belief** about probabilities before seeing data. With $\alpha = 1$, we're saying: "Before seeing data, I believe all outcomes are equally likely."
As we observe real data, this prior gets "washed out" by evidence. With 60 observations, the prior barely matters. With 6 observations, it matters more. With 0 observations, it's all we have.

This is Bayesian reasoning: prior beliefs tempered by evidence.

5.5 Worked Example: Text Classification with Smoothing

The medical examples showed *why* we need smoothing. Now let's build computational fluency with a text classification problem.

5.5.1 The Zero-Frequency Problem Demonstrated

Suppose we're building a movie review classifier with a tiny training set.
Training data:

- **Positive reviews** (2 documents): "good movie", "good good"
- **Negative reviews** (1 document): "bad movie"

Counting word occurrences:

Word	Positive (4 words)	Negative (2 words)
"good"	3 occurrences	0 occurrences
"movie"	1 occurrence	1 occurrence
"bad"	0 occurrences	1 occurrence

$V = \{\text{good, movie, bad}\}, |V| = 3$

Figure 18: *Zero counts cause the zero-frequency problem.*

Without smoothing:
For positive class (4 total words):

$$P(\text{"good"} \mid +) = \frac{3}{4} = 0.75$$
$$P(\text{"movie"} \mid +) = \frac{1}{4} = 0.25$$
$$P(\text{"bad"} \mid +) = \frac{0}{4} = \mathbf{0} \quad \leftarrow \text{Problem!}$$

For negative class (2 total words):

$$P(\text{"good"} \mid -) = \frac{0}{2} = \mathbf{0} \quad \leftarrow \text{Problem!}$$
$$P(\text{"movie"} \mid -) = \frac{1}{2} = 0.5$$
$$P(\text{"bad"} \mid -) = \frac{1}{2} = 0.5$$

Classifying "good movie": the negative score becomes $0.33 \times \mathbf{0} \times 0.5 = \mathbf{0}$. The word "good" never appeared in negative reviews, so negative class becomes impossible.

5.5.2 Applying Laplace Smoothing

With $\alpha = 1$, we add a pseudocount of 1 to every word in every class:

$$P(w \mid c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|}$$

For positive class (count(+) = 4 words, $|V| = 3$):

$$P(\text{“good"} \mid +) = \frac{3+1}{4+3} = \frac{4}{7} \approx 0.571$$

$$P(\text{“movie"} \mid +) = \frac{1+1}{4+3} = \frac{2}{7} \approx 0.286$$

$$P(\text{“bad"} \mid +) = \frac{0+1}{4+3} = \frac{1}{7} \approx 0.143 \quad \leftarrow \text{No longer zero!}$$

For negative class (count(−) = 2 words, $|V| = 3$):

$$P(\text{“good"} \mid -) = \frac{0+1}{2+3} = \frac{1}{5} = 0.200 \quad \leftarrow \text{No longer zero!}$$

$$P(\text{“movie"} \mid -) = \frac{1+1}{2+3} = \frac{2}{5} = 0.400$$

$$P(\text{“bad"} \mid -) = \frac{1+1}{2+3} = \frac{2}{5} = 0.400$$

Classify “good movie" with smoothed probabilities:
Assuming priors $P(+) = 2/3$ and $P(-) = 1/3$:
For positive:

$$\text{Score}(+) = \frac{2}{3} \times \frac{4}{7} \times \frac{2}{7} = \frac{16}{147} \approx 0.109$$

For negative:

$$\text{Score}(-) = \frac{1}{3} \times \frac{1}{5} \times \frac{2}{5} = \frac{2}{75} \approx 0.027$$

Since $0.109 > 0.027$, predict **Positive**.
Posterior probability:

$$P(+ \mid \text{“good movie"}) = \frac{0.109}{0.109 + 0.027} \approx 0.80 = 80\%$$

Smoothing Summary

Zero counts force Naive Bayes to assign *impossible* probabilities to entire classes. One unseen word can collapse a prediction.
Laplace smoothing fixes this:

$$P(w \mid c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|}$$

What this achieves:

- No more zero probabilities
- Predictions stay flexible with limited data
- Pseudocount influence fades as we collect more data

6 Extension to Continuous Features: Gaussian Naive Bayes

Up until now, we've worked with symptoms that fall into neat categories: fever is either "high" or "normal," cough is either "productive" or "dry." But real diagnosis rarely works this cleanly. When a nurse measures temperature, the reading isn't "high" or "normal"—it's 38.7°C. When monitoring heart rate, you see 95 beats per minute, not "elevated" or "normal." These *continuous* measurements carry more information than coarse categories.

6.1 The Challenge with Counting

Let's see why counting breaks down. Suppose we want to compute the likelihood of temperature 38.7°C in bacterial pneumonia cases.
Our old formula:

$$P(\text{Temperature} = 38.7 \mid \text{Bacterial}) = \frac{\#\text{ bacterial patients with temp exactly } 38.7°\text{C}}{\#\text{ bacterial patients}}$$

Looking through our 60 bacterial cases:

- Patient 1: 38.2°C
- Patient 2: 39.1°C
- Patient 3: 38.7°C ← One match!
- Patient 4: 38.71°C ← Does this count?

Only one patient has *exactly* 38.7°C. Our estimate becomes $1/60 = 0.017$—and for 38.71°C, it might be zero.
This is absurd. A temperature of 38.71°C should be treated almost identically to 38.7°C, yet counting gives wildly different estimates.

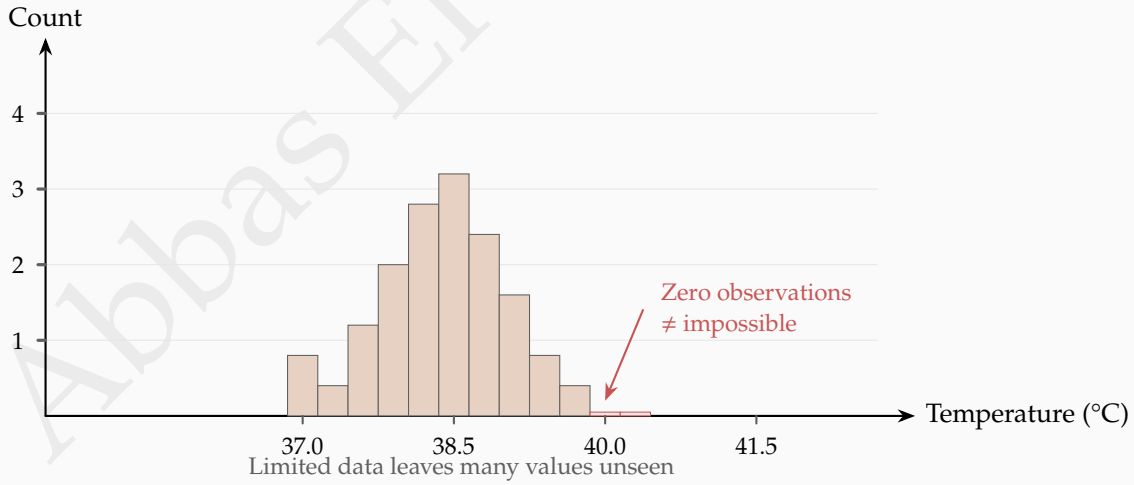


Figure 19: *A sparse temperature sample: zero counts mean unobserved—not impossible.*

6.2 From Counting to Modeling Distributions

The solution is a shift in perspective. Instead of asking "how many times did we see exactly this value?", we ask: "what *pattern* do these measurements follow?"
Look at temperature readings from our 60 bacterial patients:

38.2, 39.1, 38.7, 39.5, 38.4, 38.9, 39.2, 38.1, 38.6, 39.0, ...

These aren't random. They cluster around 38.9°C and spread out from there. Some patients run cooler (38.1°C), some hotter (39.5°C), but there's a clear center and typical spread.
Now look at the 140 viral infection cases:

$$37.1, \quad 37.8, \quad 36.9, \quad 37.5, \quad 38.2, \quad 37.3, \quad 37.0, \quad 37.9, \quad 37.4, \quad \ldots$$

Different pattern! These cluster around 37.6°C—lower than bacterial cases.

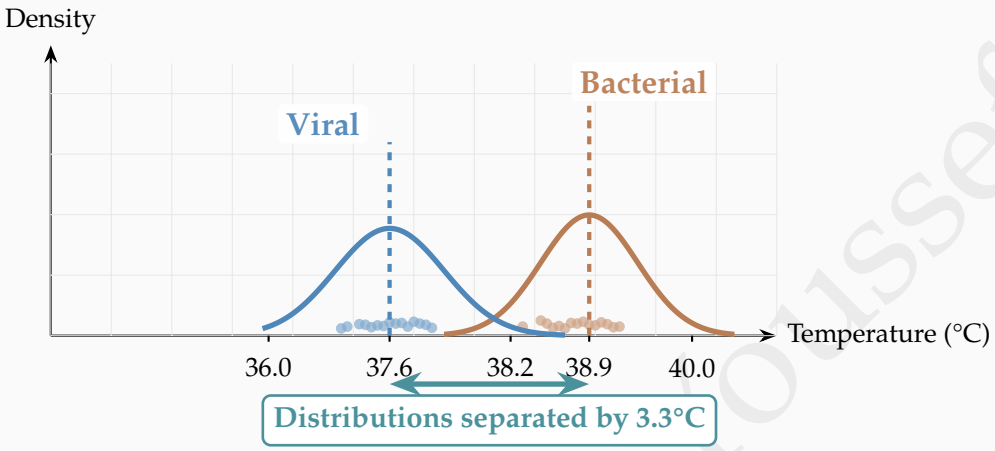


Figure 20: *Temperature distributions reveal distinct patterns. Fitting Gaussian curves captures these patterns, allowing likelihood estimation for any temperature.*

The key insight: we can *fit a curve* to each set of observations. This curve tells us how likely any temperature value is—even values we've never observed.

6.3 The Gaussian Distribution

What shape should these curves take? For many biological measurements, a bell-shaped curve called the **Gaussian** (or **normal**) distribution provides an excellent fit.
Why bell-shaped? A patient's temperature is influenced by many small factors: baseline metabolism, hydration, time of day, exact stage of infection, measurement precision. When many independent factors combine additively, the result tends toward a bell curve (the Central Limit Theorem).
A Gaussian is completely described by two numbers:

- The **mean** (μ): Where is the center?
- The **standard deviation** (σ): How spread out are values?

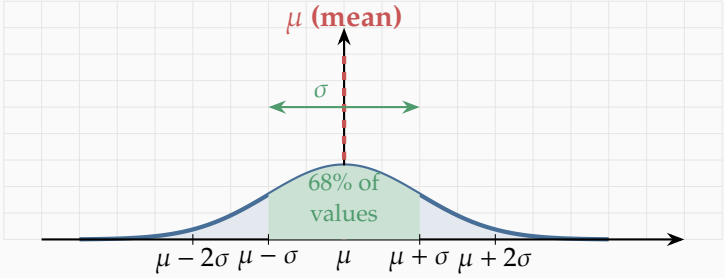


Figure 21: *The Gaussian distribution: μ sets the center, σ controls width. About 68% of values fall within one standard deviation of the mean.*

For our medical diagnosis:

Diagnosis	Mean (μ)	Std Dev (σ)	Interpretation
Bacterial	38.9°C	0.8°C	Higher temps, tighter clustering
Viral	37.6°C	0.9°C	Lower temps, more variation

These parameters are easy to estimate—just sample mean and standard deviation within each class:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \hat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2}$$

6.4 Computing the Density

The Gaussian density formula:

$$p(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

The intuition is simple:

- $(x-\mu)^2$: How far from center? Farther = lower density.
- Division by σ^2: Scaled by spread. Wide distributions penalize distance less.
- Exponential decay: Distance from center rapidly decreases density.

Temperature 38.5°C: Which Diagnosis?

Bacterial parameters: $\mu = 38.9$C, $\sigma = 0.8$C
Distance from center: $38.5 - 38.9 = -0.4$ (half a standard deviation away)

$$p(38.5 \mid \text{Bacterial}) = \frac{1}{\sqrt{2\pi(0.8)^2}} \exp\left(-\frac{(-0.4)^2}{2(0.8)^2}\right) = 0.44$$

Viral parameters: $\mu = 37.6$C, $\sigma = 0.9$C
Distance from center: $38.5 - 37.6 = 0.9$ (one full standard deviation away)

$$p(38.5 \mid \text{Viral}) = \frac{1}{\sqrt{2\pi(0.9)^2}} \exp\left(-\frac{(0.9)^2}{2(0.9)^2}\right) = 0.27$$

Likelihood ratio: $\frac{0.44}{0.27} = 1.6$
Temperature 38.5°C is 1.6 times more consistent with bacterial than viral infection.

6.5 Density vs Probability

Those density values (0.44, 0.27) aren't probabilities in the usual sense. For continuous measurements, the probability of any *exact* value is technically zero—there are infinitely many possible temperatures.
What densities tell us is the *relative* likelihood of different values.

Thinking About Density

Imagine pouring observations from each diagnosis into a container. Density at any point tells you how "crowded" that region is.

- High density (tall curve): Many observations cluster here. Typical for this class.
- Low density (short curve): Few observations here. Unusual for this class.

For classification, we only compare densities between classes. Whether 0.44 vs 0.27, or 4.4 vs 2.7—the ratio is the same, and bacterial wins.

6.6 Mixing Continuous and Discrete Features

Real patients have both types of measurements. Naive Bayes handles this naturally—multiply discrete probabilities with continuous densities.

Complete Patient with Mixed Features

Patient presents with:

- Fever status: High (discrete)
- Cough type: Productive (discrete)
- Temperature: 38.5°C (continuous)
- Heart rate: 95 bpm (continuous)

Learned parameters:

Feature	Type	Bacterial	Viral
Prior	—	0.30	0.70
Fever = High	Discrete	0.850	0.300
Cough = Productive	Discrete	0.867	0.479
Temperature	Gaussian	$\mathcal{N}(38.9, 0.64)$	$\mathcal{N}(37.6, 0.81)$
Heart Rate	Gaussian	$\mathcal{N}(98, 144)$	$\mathcal{N}(88, 225)$

Computing scores:
For bacterial:

$$0.30 \times 0.850 \times 0.867 \times \underbrace{0.44}_{\text{temp}} \times \underbrace{0.031}_{\text{HR}} = 0.0030$$

For viral:

$$0.70 \times 0.300 \times 0.479 \times \underbrace{0.27}_{\text{temp}} \times \underbrace{0.020}_{\text{HR}} = 0.00054$$

Prediction: Bacterial (score is 5.6× higher)
Posterior: $\dfrac{0.0030}{0.0030 + 0.00054} = 85\%$

6.7 Complete Walkthrough: Mixed Feature Classification

We've seen Gaussian modeling in isolation. Now let's work through a complete example that combines discrete features, continuous features, and smoothing—everything we've learned so far in one realistic scenario.

Predicting Tax Bracket from Demographics

The scenario: A tax consulting firm wants to predict whether clients will fall into a higher tax bracket based on demographic information collected during initial consultation.
Features available:

- **Marital Status** (discrete): Single, Married, or Divorced
- **Annual Income** (continuous): measured in thousands of dollars

Target classes:

- **Higher** — subject to higher tax rates
- **Standard** — standard tax bracket

Training data ($m = 10$ historical clients):

Client	Marital Status	Income ($k)	Tax Bracket
1	Single	125	Standard
2	Married	100	Standard
3	Single	70	Standard
4	Married	120	Standard
5	Divorced	95	Higher
6	Divorced	60	Standard
7	Single	220	Standard
8	Married	85	Higher
9	Divorced	75	Standard
10	Single	90	Higher

Table 1: *Historical client data with confirmed tax bracket classifications.*

Task: Classify a new client who is Single with an income of $85k.

STEP 1: Estimate Class Priors
Count the class occurrences:

- Higher bracket: 3 clients (rows 5, 8, 10)
- Standard bracket: 7 clients (rows 1, 2, 3, 4, 6, 7, 9)
- Total: $m = 10$

With Laplace smoothing ($\alpha = 1$, $K = 2$ classes):

$$P(\text{Higher}) = \frac{3+1}{10+2} = \frac{4}{12} = 0.333$$

$$P(\text{Standard}) = \frac{7+1}{10+2} = \frac{8}{12} = 0.667$$

STEP 2: Estimate Likelihoods for Marital Status (Discrete Feature)

For the **Higher** bracket ($m_{\text{Higher}} = 3$ clients):
Counts: Single (1), Married (1), Divorced (1)
With smoothing ($\alpha = 1$, $d = 3$ marital statuses):

$$P(\text{Single} \mid \text{Higher}) = \frac{1+1}{3+3} = \frac{2}{6} = 0.333$$
$$P(\text{Married} \mid \text{Higher}) = \frac{1+1}{3+3} = \frac{2}{6} = 0.333$$
$$P(\text{Divorced} \mid \text{Higher}) = \frac{1+1}{3+3} = \frac{2}{6} = 0.333$$

For the **Standard** bracket ($m_{\text{Standard}} = 7$ clients):
Counts: Single (3), Married (2), Divorced (2)

$$P(\text{Single} \mid \text{Standard}) = \frac{3+1}{7+3} = \frac{4}{10} = 0.400$$
$$P(\text{Married} \mid \text{Standard}) = \frac{2+1}{7+3} = \frac{3}{10} = 0.300$$
$$P(\text{Divorced} \mid \text{Standard}) = \frac{2+1}{7+3} = \frac{3}{10} = 0.300$$

STEP 3: Estimate Parameters for Income (Continuous Feature)
For the **Higher** bracket, incomes are: 95, 85, 90 (in thousands)
Mean:

$$\mu_{\text{Higher}} = \frac{95 + 85 + 90}{3} = 90.0$$

Sample variance:

$$\sigma^2_{\text{Higher}} = \frac{(95-90)^2 + (85-90)^2 + (90-90)^2}{3-1} = \frac{25 + 25 + 0}{2} = 25.0$$

Standard deviation: $\sigma_{\text{Higher}} = 5.0$

For the **Standard** bracket, incomes are: 125, 100, 70, 120, 60, 220, 75
Mean:

$$\mu_{\text{Standard}} = \frac{125 + 100 + 70 + 120 + 60 + 220 + 75}{7} = 110.0$$

Sample variance:

$$\begin{aligned}\sigma^2_{\text{Standard}} &= \frac{(125-110)^2 + (100-110)^2 + (70-110)^2 + \cdots + (75-110)^2}{7-1} \\ &= \frac{225 + 100 + 1600 + 100 + 2500 + 12100 + 1225}{6} \\ &= \frac{17850}{6} = 2975.0\end{aligned}$$

Standard deviation: $\sigma_{\text{Standard}} \approx 54.5$

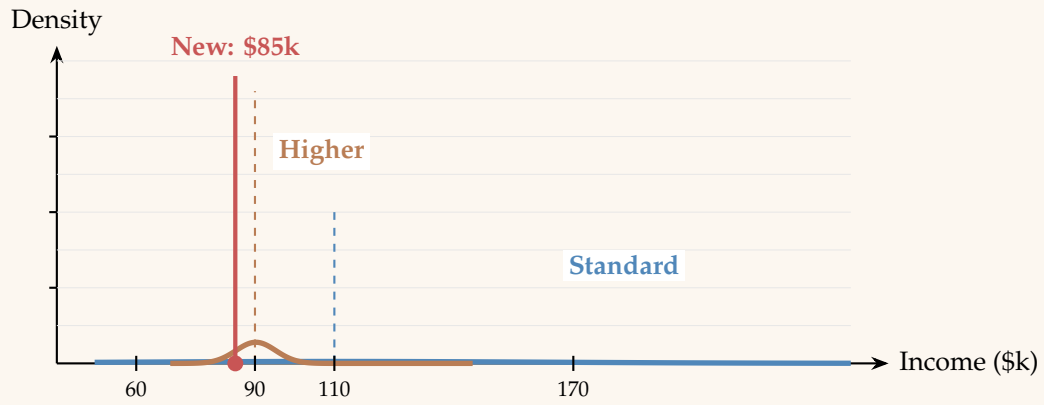


Figure 22: *Income distributions for each tax bracket. The Higher bracket shows tight clustering around $90k, while Standard has a much wider spread around $110k. The new client's $85k income falls very close to the Higher bracket's mean.*

STEP 4: Compute Gaussian Densities for the New Client
New client has income = $85k. We evaluate the density at this point for each class.
For **Higher** ($\mu = 90$, $\sigma = 5.0$):

$$\begin{aligned} p(85 \mid \text{Higher}) &= \frac{1}{\sqrt{2\pi \cdot 25}} \exp\left(-\frac{(85-90)^2}{2 \cdot 25}\right) \\ &= \frac{1}{\sqrt{157.08}} \exp\left(-\frac{25}{50}\right) \\ &= 0.0798 \times 0.6065 \\ &\approx 0.0484 \end{aligned}$$

For **Standard** ($\mu = 110$, $\sigma = 54.5$):

$$\begin{aligned} p(85 \mid \text{Standard}) &= \frac{1}{\sqrt{2\pi \cdot 2975}} \exp\left(-\frac{(85-110)^2}{2 \cdot 2975}\right) \\ &= \frac{1}{\sqrt{18691.2}} \exp\left(-\frac{625}{5950}\right) \\ &= 0.00731 \times 0.9003 \\ &\approx 0.00658 \end{aligned}$$

Notice the density ratio: $0.0484/0.00658 \approx 7.4$. Income of $85k is 7.4 times more consistent with the Higher bracket than Standard—strong evidence despite being slightly below the Higher mean.

STEP 5: Compute Posterior Scores
Combine everything: prior × discrete likelihood × continuous density.
For **Higher**:

$$\begin{aligned} \text{Score}(\text{Higher}) &= P(\text{Higher}) \times P(\text{Single} \mid \text{Higher}) \times p(85 \mid \text{Higher}) \\ &= 0.333 \times 0.333 \times 0.0484 \end{aligned}$$

$$\approx 0.00537$$

For **Standard**:

$$\begin{aligned}\text{Score}(\text{Standard}) &= P(\text{Standard}) \times P(\text{Single} \mid \text{Standard}) \times p(85 \mid \text{Standard}) \\ &= 0.667 \times 0.400 \times 0.00658 \\ &\approx 0.00176\end{aligned}$$

STEP 6: Make the Decision
Compare the scores:

$$0.00537 > 0.00176$$

Prediction: Higher tax bracket
Normalizing to obtain posterior probability:

$$P(\text{Higher} \mid \mathbf{x}) = \frac{0.00537}{0.00537 + 0.00176} = \frac{0.00537}{0.00713} \approx 0.753 = 75.3\%$$

Understanding the prediction:

- The client's income ($85k) is only $5k from the Higher bracket mean ($90k)—exactly one standard deviation
- It's $25k below the Standard bracket mean ($110k)
- The Gaussian density at $85k is 7.4× higher under Higher than Standard
- This strong evidence from income outweighs the moderately higher prior for Standard (67% vs 33%)
- Marital status is roughly uniformly distributed across both classes (all ≈ 0.33), so it contributes minimal information

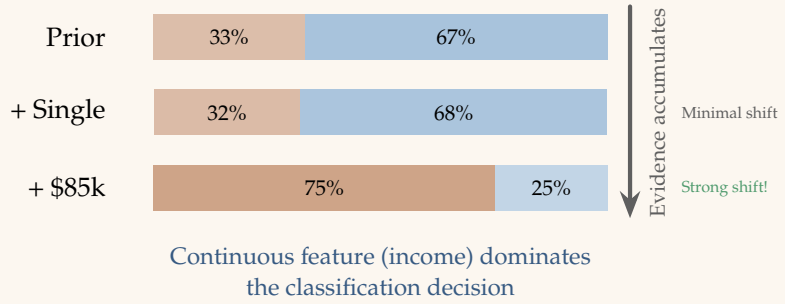


Figure 23: *How evidence shifts beliefs: marital status provides little discriminatory power (roughly uniform across classes), but income strongly favors the Higher bracket. The final posterior reflects the accumulation of all evidence, weighted by diagnostic value.*

This example demonstrates the complete Naive Bayes workflow with mixed feature types: counting for priors, Laplace smoothing for discrete features, Gaussian parameter estimation for continuous features, and principled combination of all evidence types.
The result is fully interpretable—we can trace exactly how each feature contributed to the

final classification and quantify the strength of evidence each provides. This interpretability, combined with computational efficiency and strong empirical performance, explains why Naive Bayes remains a workhorse algorithm in production machine learning systems.

Gaussian Naive Bayes Summary

For continuous features:

1. Fit a Gaussian to each feature within each class (estimate μ and σ)
2. For new observations, compute density under each class's distribution
3. Use densities exactly like discrete probabilities—multiply them together

Key insight: We're not asking "how often did we see exactly this value?" but rather "how consistent is this value with the typical pattern for each class?"
When it works well: Features roughly bell-shaped within each class
When it struggles: Heavily skewed or multi-modal distributions

7 Numerical Stability: Log-Space Computation

Our classifier works beautifully in theory. But there's a practical problem when implementing on a computer: scores can become vanishingly small.

7.1 The Underflow Problem

Look at scores from our last example:

- Bacterial: 0.0030
- Viral: 0.00054

Already tiny with just 4 features! What happens with 20 features?
Suppose each probability is around 0.5:

$$\text{Score} = 0.3 \times 0.5^{20} = 0.3 \times 9.5 \times 10^{-7} = 2.86 \times 10^{-7}$$

With 50 features:

$$\text{Score} = 0.3 \times 0.5^{50} = 2.67 \times 10^{-16}$$

With 100 features:

$$\text{Score} = 0.3 \times 0.5^{100} = 2.37 \times 10^{-31}$$

Underflow Catastrophe

Computer's limit: Most systems can't represent numbers smaller than $\approx 10^{-308}$.
Beyond 300 features: Numbers become exactly zero (underflow).
Result: All classes score zero. Classifier breaks completely.
This happens with real datasets having hundreds or thousands of features (text classification, gene expression, etc.).

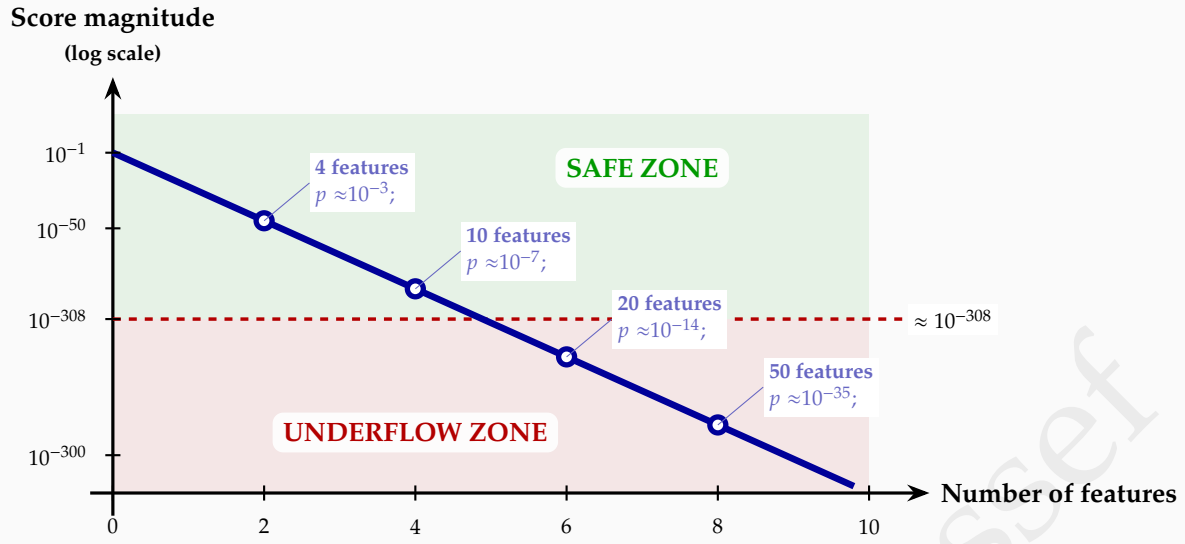


Figure 24: *As features increase, scores decrease exponentially. Beyond* $\approx 10^{-308}$*, values underflow to zero.*

7.2 The Logarithm Solution

Key insight: logarithms convert products into sums.
Key properties:

1. $\log(a \times b) = \log(a) + \log(b)$ — Product becomes sum
2. If $a > b$, then $\log(a) > \log(b)$ — Preserves ordering
3. $\log(10^{-100}) = -230$ — Tiny numbers become moderate negatives

Why Logarithms Save Us

Instead of multiplying tiny probabilities:

$$p_1 \times p_2 \times p_3 \times \cdots \times p_{100} \quad \text{(underflows!)}$$

We add their logarithms:

$$\log p_1 + \log p_2 + \log p_3 + \cdots + \log p_{100} \quad \text{(safe!)}$$

Result: Instead of 10^{-200} (impossible), we get -460 (perfectly representable).
For classification: Since log preserves ordering, arg max of scores = arg max of log-scores.
Note: log denotes natural logarithm, though any base yields identical predictions.

7.3 Log-Space Naive Bayes

Original rule:

$$\hat{y} = \arg\max_c \left[P(Y = c) \prod_{j=1}^{n} P(x_j \mid Y = c) \right]$$

Log-space equivalent:

$$\hat{y} = \arg\max_c \left[\log P(Y = c) + \sum_{j=1}^{n} \log P(x_j \mid Y = c) \right]$$

Same prediction, but all computations stay in a safe numerical range.

7.4 Log-Gaussian Density

For continuous features with Gaussian distributions:

$$\log p(x) = -\frac{1}{2} \log(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2}$$

No exponentials, no divisions by tiny numbers—just logarithms and arithmetic.

7.5 Complete Log-Space Example

Same Patient, Log-Space Calculation

Patient features:

- Fever: High (discrete)
- Cough: Productive (discrete)
- Temperature: 38.5°C (continuous)
- Heart rate: 95 bpm (continuous)

Bacterial log-score:

$$\begin{aligned} \log \text{Score}_{\text{Bact}} &= \log(0.30) + \log(0.850) + \log(0.867) \\ &\quad + \log p(38.5 \mid 38.9, 0.8) + \log p(95 \mid 98, 12) \\ &= -1.204 + (-0.163) + (-0.143) + (-0.826) + (-3.434) \\ &= -5.770 \end{aligned}$$

Viral log-score:

$$\begin{aligned} \log \text{Score}_{\text{Viral}} &= \log(0.70) + \log(0.300) + \log(0.479) \\ &\quad + \log p(38.5 \mid 37.6, 0.9) + \log p(95 \mid 88, 15) \\ &= -0.357 + (-1.204) + (-0.735) + (-1.234) + (-3.647) \\ &= -7.177 \end{aligned}$$

Comparison:

$$-5.770 > -7.177 \implies \text{Predict Bacterial}$$

Probability (if needed):

$$P(\text{Bact} \mid \mathbf{x}) = \frac{e^{-5.770}}{e^{-5.770} + e^{-7.177}} = \frac{0.00308}{0.00308 + 0.00076} = 80.2\%$$

Same answer, numerically stable for any number of features.

Production-Ready Log-Space Algorithm

Training Phase: (Same as before)

1. Compute priors: $P(Y = c)$
2. For discrete features: $P(X_j = v \mid Y = c)$
3. For continuous features: μ_{jc}, σ^2_{jc}

Prediction Phase (for new instance **x**):
For each class c**:**

1. Initialize: $\text{logScore}(c) = \log P(Y = c)$
2. For each discrete feature j:

$$\text{logScore}(c) \mathrel{+}= \log P(x_j \mid Y = c)$$

3. For each continuous feature j:

$$\text{logScore}(c) \mathrel{+}= -\frac{1}{2}\log(2\pi\sigma^2_{jc}) - \frac{(x_j - \mu_{jc})^2}{2\sigma^2_{jc}}$$

Predict: $\hat{y} = \arg\max_c \text{logScore}(c)$
If probabilities needed:

$$P(Y = c \mid \mathbf{x}) = \frac{\exp(\text{logScore}(c))}{\sum_{c'} \exp(\text{logScore}(c'))}$$

Log-Space Summary

Why use log-space:

- **Prevents underflow:** Works with any number of features
- **Faster:** Addition is faster than multiplication
- **More accurate:** Avoids floating-point error accumulation
- **Standard practice:** All production ML libraries use this

Implementation checklist:

1. Train in regular space (counts, means, variances)
2. Convert to log-space for prediction
3. Never exponentiate until final probability (if needed)
4. For classification only: Skip exponentiation—just compare log-scores

Common mistake: Computing probabilities first, then taking log. Underflow happens before you apply log!
Correct approach: Compute logs directly from parameters; never form tiny products.

8 The Surprising Success: Why Naive Bayes Works Despite Being Wrong

We've built a complete probabilistic classifier. We can handle discrete and continuous features, avoid zero probabilities through smoothing, and compute safely in log-space.
But we've been carrying an uncomfortable truth: the fundamental assumption behind Naive Bayes is almost always wrong.
We assumed features are conditionally independent given the class. In our medical example: once you know a patient has bacterial pneumonia, learning they have high fever tells you nothing about whether they'll have a productive cough.
That's not how biology works. Fever and cough are both influenced by infection severity and location. They're correlated even within each diagnosis.
So why does Naive Bayes work so well in practice?

8.1 What Classification Actually Requires

The answer lies in understanding what classification needs versus what probability modeling needs.
For classification: We need to know which class has the higher posterior probability.
We do NOT need: The exact numerical values of those probabilities.

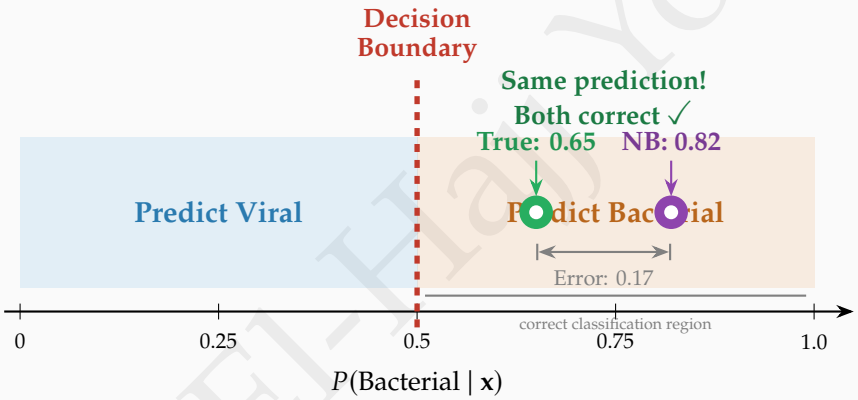


Figure 25: *True probability is 0.65, Naive Bayes estimates 0.82. Despite 0.17 error, both land on the same side of 0.5—yielding identical predictions.*

Classification only cares about which side of the decision boundary we land on. Probability estimates can be miscalibrated—too confident, too uncertain, systematically biased—and we'll still classify correctly as long as the *ranking* is preserved.
Naive Bayes often gets probabilities wrong but rankings right. That's enough for classification.

8.2 When Independence Violations Cancel Out

Not all violations of independence are equally harmful.
Scenario A: Symmetric correlation
Suppose fever and chills are correlated in both bacterial and viral cases (both are temperature-regulation symptoms).

- In bacterial cases: fever makes chills more likely
- In viral cases: fever also makes chills more likely
- The correlation is roughly the same in both classes

What happens? Naive Bayes treats fever and chills as independent evidence when they're really echoes of the same signal. It overcounts the evidence—but overcounts *equally for both classes*. Scores are both inflated, but their ratio stays roughly correct.

Scenario B: Asymmetric correlation

Now suppose symptoms A and B have:

- In bacterial cases: A and B always occur together (perfect correlation)
- In viral cases: A and B are independent

This is trouble. Naive Bayes double-counts evidence for bacterial (treating A and B as separate when they're redundant), but counts correctly for viral. The bacterial score gets artificially inflated, potentially flipping the decision.

Symmetric Correlation

Both classes: AB correlated

Effect: Both scores inflated

Ratio: Approximately preserved

Classification: Usually correct

Asymmetric Correlation

Class 1: AB correlated
Class 2: AB independent

Effect: One score inflated more

Ratio: Distorted

Classification: May fail

Figure 26: *Symmetric correlations tend to cancel out; asymmetric ones can cause errors.*

Fortunately, many real-world correlations are roughly symmetric—features correlate due to underlying mechanisms affecting both classes similarly.

8.3 The Geometry: Linear Decision Boundaries

Another perspective explains Naive Bayes's robustness. Taking logarithms of the classification rule:

$$\log P(Y = c \mid \mathbf{x}) = \log P(Y = c) + \sum_{j=1}^{n} \log P(x_j \mid Y = c) + \text{const}$$

This is a weighted sum of individual feature contributions—a *linear* function. The decision boundary between classes is a hyperplane in feature space.

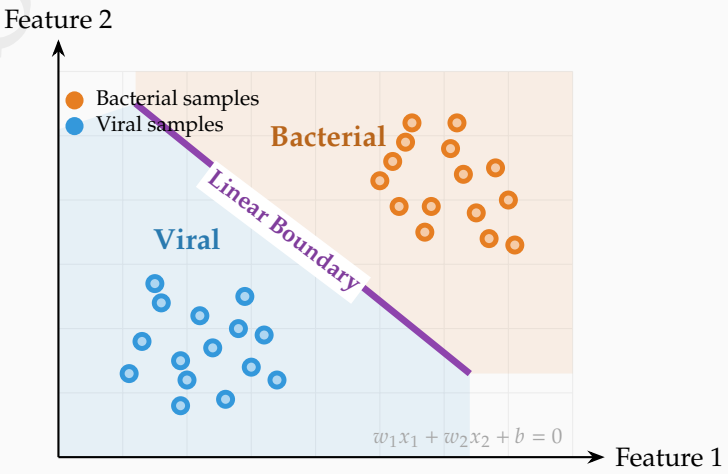


Figure 27: *Naive Bayes produces a linear decision boundary. When classes are approximately linearly separable, it performs well.*

Even when probability estimates are wrong, the classifier learns a reasonable linear separator. It may not find the *optimal* boundary, but often finds one that works.
Many real classification problems have roughly linearly separable classes, and Naive Bayes finds a decent boundary regardless of how well it models the true distribution.

8.4 The Data Efficiency Advantage

One more piece to the puzzle: how much data each approach needs.
With 20 binary features, a "full" model capturing all feature interactions needs $2^{20} \approx 1$ million parameters. Naive Bayes needs only $2 \times 20 = 40$.

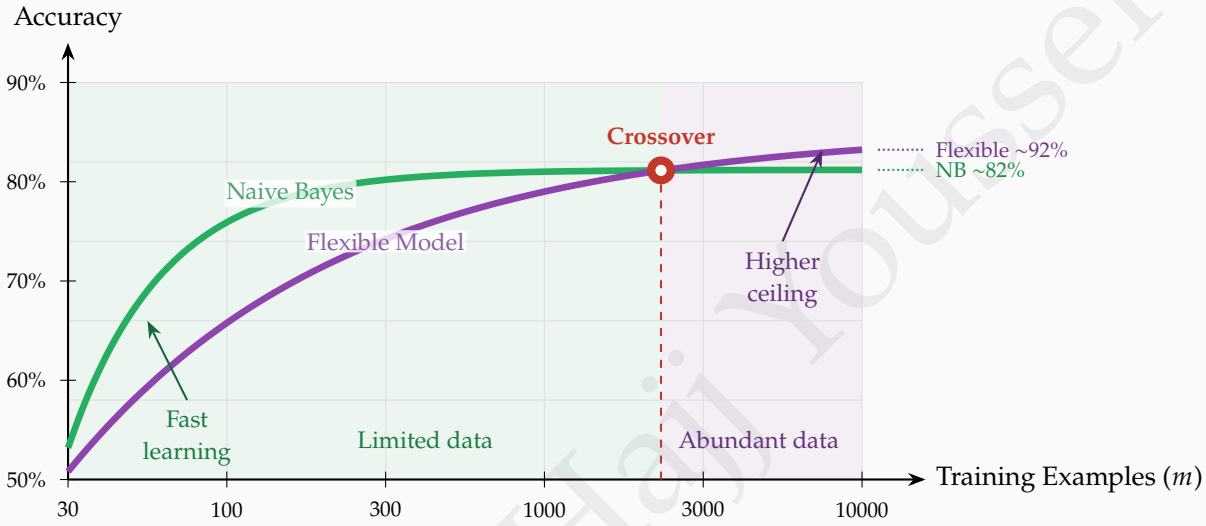


Figure 28: *Learning curves: Naive Bayes converges quickly but plateaus early. Flexible models need more data but achieve higher asymptotic accuracy.*

With limited data:

- Flexible models can't reliably estimate many parameters; they overfit
- Naive Bayes quickly converges; its bias keeps it stable

With abundant data:

- Flexible models leverage their expressiveness
- Naive Bayes hits a ceiling—its assumptions limit fit quality

The crossing point depends on the problem. In text classification (high-dimensional, sparse features), Naive Bayes often remains competitive even with substantial data. In problems with strong feature interactions and dense data, flexible methods eventually dominate.

When to Trust Naive Bayes

Naive Bayes excels when:

- Data is limited relative to feature count
- Feature correlations are roughly symmetric across classes
- Fast training and prediction matter
- You need a robust baseline

- Features are roughly independent or weakly correlated

Consider alternatives when:

- You have abundant data
- Features have complex, class-specific interactions
- Calibrated probabilities are crucial (not just rankings)
- The true decision boundary is highly non-linear

The "naive" assumption is a feature, not a bug—it trades modeling accuracy for stability and data efficiency. For many problems, that's a good trade.

9 Beyond Independence: Bayesian Networks

Throughout this chapter, we've made peace with a simplifying assumption: features are independent given the class. This works surprisingly well, but it's frequently wrong.
What if we could have both—computational tractability of probabilistic reasoning with flexibility to model dependencies we know exist?
This is what **Bayesian Networks** offer.

9.1 The Middle Ground

Think of modeling options as a spectrum:

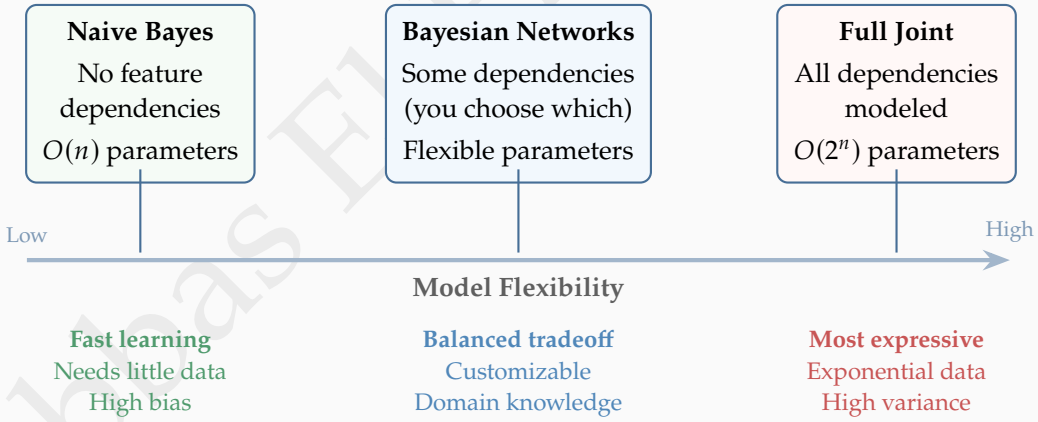


Figure 29: *Bayesian Networks sit between extremes, modeling dependencies that matter while ignoring those that don't.*

Naive Bayes assumes everything is independent, estimates few parameters, works with minimal data. The full joint models every interaction, needs astronomical data.
Bayesian Networks let you choose your position. Know that two symptoms are mechanistically linked? Model that dependency. Believe two others are independent? Leave them unconnected. You encode domain knowledge into structure, and the math follows.

9.2 A Simple Example: Weather and Decisions

Setting: You wake up and notice your lawn is wet. Did it rain last night, or did someone leave the sprinkler on?

Variables:

- C: Is it cloudy?
- S: Was the sprinkler on?
- R: Did it rain?
- W: Is the grass wet?

What influences what?

- Cloudy weather makes rain more likely
- On cloudy days, you're less likely to run the sprinkler
- Both rain and sprinkler can make grass wet

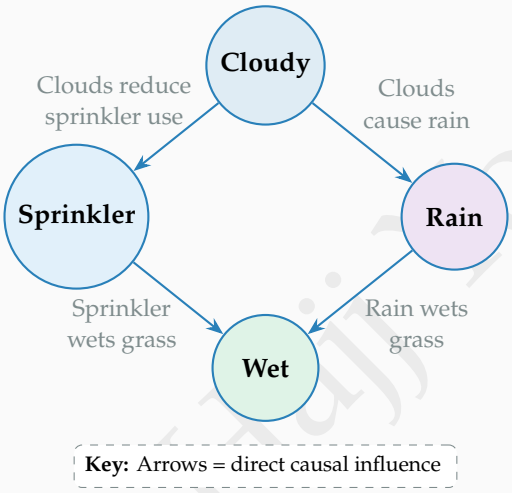


Figure 30: *The "Wet Grass" Bayesian Network. Arrows show direct causal influences.*

9.3 What the Graph Tells Us

The structure encodes independence assumptions.
Key principle: A node is independent of non-descendants once you know its parents.
Sprinkler and Rain: No direct arrow between them. Are they independent?
Not unconditionally! Both are influenced by Cloudy. On cloudy days, rain is likely and sprinkler unlikely. On sunny days, the reverse. Knowing it rained tells you something about the sprinkler.
But *given* whether it's cloudy—once we condition on their common parent—Sprinkler and Rain become independent. Knowing it's cloudy, learning that it rained doesn't change your belief about the sprinkler.
Wet Grass and Cloudy: No direct arrow. Does cloudiness affect wetness?
Indirectly—through Rain and Sprinkler. But if you know whether it rained and whether the sprinkler ran, cloudiness tells you nothing additional.
The graph makes these independence relationships visual and explicit.

9.4 The Numbers Behind the Graph

A graph alone doesn't let us compute probabilities—we also need numbers. For each node, we specify a **conditional probability table (CPT)**: "Given what my parents are doing, here are my probabilities."

$P(C)$	
C	Prob
Yes	0.5
No	0.5

$P(S \mid C)$	
C	$P(S{=}\text{On})$
Yes	0.10
No	0.50

$P(R \mid C)$	
C	$P(R{=}\text{Yes})$
Yes	0.80
No	0.20

$P(W \mid S, R)$		
S	R	$P(W{=}\text{Yes})$
Off	No	0.00
Off	Yes	0.90
On	No	0.90
On	Yes	0.99

Table 2: *Conditional probability tables for the Wet Grass network.*

Notice the structure:

- **Cloudy** has no parents—just a prior probability
- **Sprinkler** depends only on Cloudy
- **Rain** depends only on Cloudy
- **Wet** depends on both Sprinkler and Rain

The joint probability factors according to the graph:

$$P(C, S, R, W) = P(C) \times P(S \mid C) \times P(R \mid C) \times P(W \mid S, R)$$

Each variable's probability depends only on its parents.

9.5 A Surprising Phenomenon: Explaining Away

Bayesian Networks reveal reasoning patterns that seem counterintuitive but are deeply logical.
You observe wet grass.
Initial reasoning: Two possible explanations—rain or sprinkler. Wet grass makes both more plausible.
Now suppose you learn the sprinkler was definitely on.
Updated reasoning: You have an explanation for wet grass. Does this change your belief about rain?
Yes—your belief that it rained goes *down*.

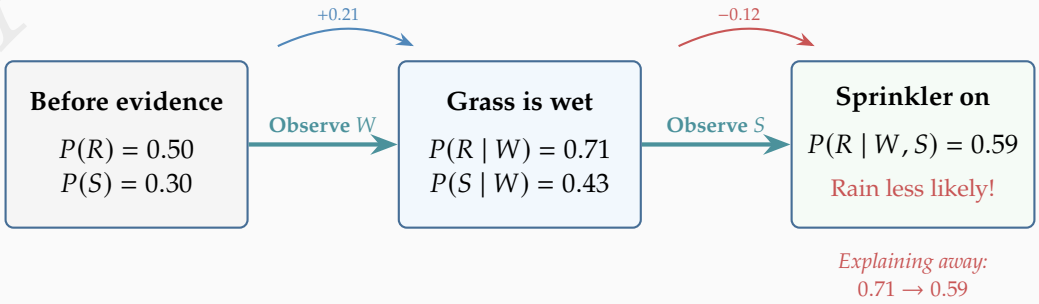


Figure 31: *Explaining away: Confirming one cause (sprinkler) reduces probability of alternative causes (rain).*

This is **explaining away**. Once the sprinkler "explains" the wet grass, there's less need to invoke rain. The two causes compete to explain the effect.
This pattern appears everywhere:

- Learning a patient has Disease A makes Disease B (causing the same symptom) less likely
- Learning a student studied hard makes "test was easy" less likely
- Learning the battery is dead makes "out of gas" less likely

Naive Bayes cannot capture this—it treats all features as independent evidence. Bayesian Networks can, because they model causal structure.

9.6 Connecting Back to Naive Bayes

Where does Naive Bayes fit? It's a special case—a Bayesian Network with specific, simple structure.

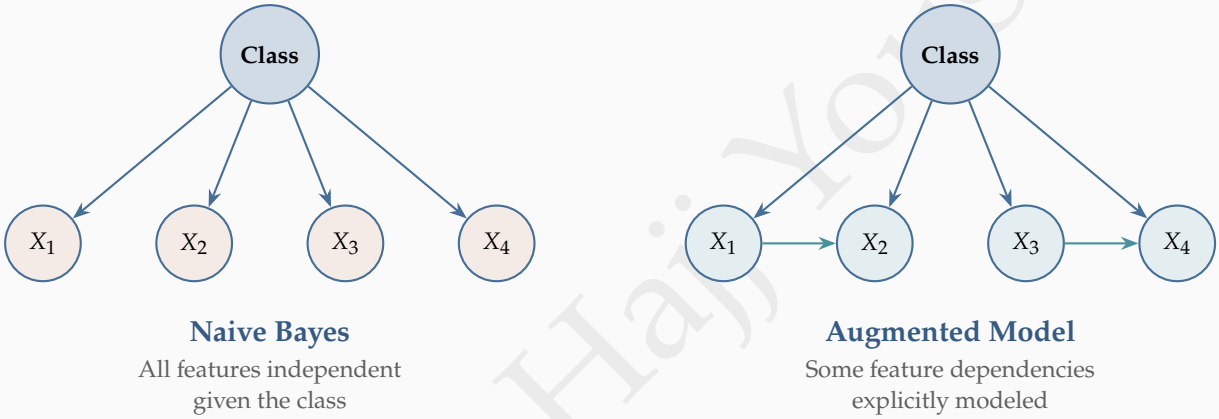


Figure 32: *Naive Bayes is a Bayesian Network where only the class has children. Adding edges between features models dependencies.*

In Naive Bayes, the class is the only parent of any feature—forcing conditional independence. Adding an edge from X_1 to X_2 says: "Even knowing the class, X_1 tells us something about X_2." The factorization changes:

$$P(X_1, X_2 \mid Y) = P(X_1 \mid Y) \times P(X_2 \mid X_1, Y)$$

Now X_2's probability depends on X_1, not just the class. You've modeled a dependency.

9.7 The Price of Flexibility

More edges mean more expressive power—but also more parameters.

Model	Parameters ($n = 10$ binary features)	Tradeoff
Naive Bayes	20	Ignores all feature dependencies
Tree-Augmented	30	Each feature can depend on one other
Full pairwise	200	All pairs modeled, expensive
Full joint	1,024	All interactions, often impractical

The art of Bayesian Networks is choosing the right structure—modeling dependencies that matter while keeping the network tractable.

Domain knowledge is crucial. A physician knows certain symptoms cluster together. A spam filter designer knows certain words co-occur in specific contexts. This knowledge guides which edges to include.

Bayesian Networks: The Big Picture

What they offer:

- Graphical language for expressing probabilistic dependencies
- Flexibility to model some relationships while assuming others independent
- Principled inference: compute any probability query from structure
- Capture phenomena like explaining away that simpler models miss

Relationship to Naive Bayes:

- Naive Bayes is a Bayesian Network with star structure (class $\rightarrow$ all features)
- Adding edges relaxes independence assumption
- More edges = more expressive but needs more data

When to consider them:

- You have domain knowledge about which variables influence others
- Independence assumption hurts performance
- You need complex probabilistic queries, not just classification
- Interpretability matters: graph shows modeling assumptions clearly

10 Bringing It All Together

We began with a physician facing uncertainty at 3 AM. Multiple symptoms, none definitive, and a decision that couldn't wait. This is probabilistic classification: combining imperfect evidence to make the best possible choice.

10.1 The Journey We've Taken

The challenge: Real-world classification involves multiple features, and memorizing every combination is impossible. With $n = 20$ binary features, there are over a million combinations—far more than any dataset could cover.
The insight: Instead of learning complete patterns, learn how each feature behaves within each class, then combine evidence using probability theory.
The mechanics:

1. **Priors** tell us baseline class rates
2. **Likelihoods** tell us how features behave within each class
3. **Bayes' theorem** combines them into posteriors
4. **Classification** picks the highest-posterior class

The assumption: Features are conditionally independent given the class. Often wrong—but the simplicity it buys is worth the price, especially with limited data.
Practical necessities:

- **Smoothing** prevents zero probabilities from derailing predictions
- **Log-space computation** prevents numerical underflow
- **Gaussian modeling** extends to continuous measurements

The bigger picture: Naive Bayes is a special case of Bayesian Networks, which offer a graphical language for encoding probabilistic relationships. When independence is too restrictive, these flexible models capture dependencies—at the cost of needing more data.

10.2 What Makes Naive Bayes Special

Among classification methods, Naive Bayes occupies a unique position:

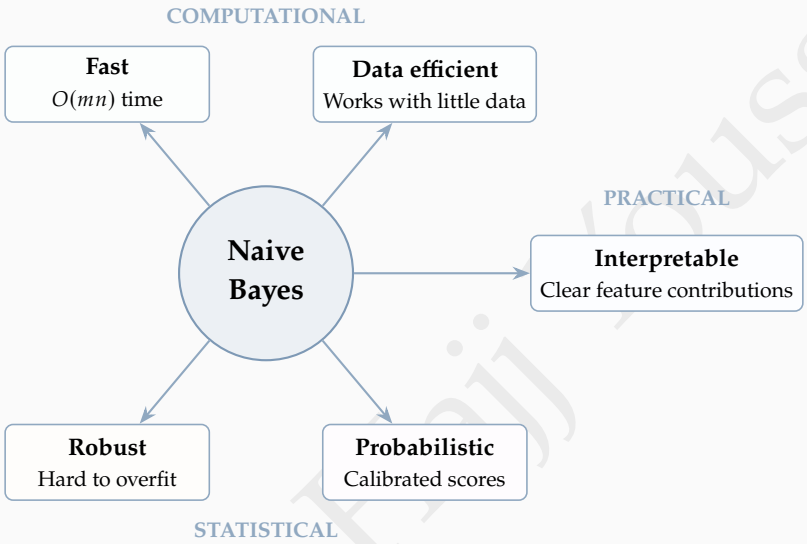


Figure 33: *Naive Bayes combines computational, statistical, and practical advantages rarely found together.*

It trains in a single pass—just count and compute means. It produces probability estimates, not just class labels. Predictions are interpretable: you can see exactly how each feature contributed. Strong assumptions act as regularization, preventing overfitting even with high-dimensional, sparse data.

These properties make Naive Bayes an excellent first model for any classification problem. If it works well, you may not need anything more complex. If it doesn't, you've established a baseline and learned something about your data's structure.

10.3 Practical Guidance

When building a Naive Bayes classifier:

1. **Start with exploration.** Before modeling, examine your data. Which features differ most between classes? Are there obvious dependencies?
2. **Choose the right variant.** Bernoulli for binary features, Multinomial for counts (like word frequencies), Gaussian for continuous measurements. You can mix these within a single model.
3. **Always use smoothing.** Laplace smoothing ($\alpha = 1$) is standard. Consider smaller values ($\alpha = 0.1$) with abundant data.

4. **Work in log space.** Every production implementation does this. It's necessary for numerical stability.
5. **Validate thoughtfully.** Naive Bayes can give overconfident probability estimates. If calibrated probabilities matter (not just rankings), consider Platt scaling or isotonic regression.
6. **Examine errors.** When Naive Bayes fails, errors often reveal structure: correlated features, rare classes with insufficient data, or non-linear boundaries it cannot capture.

10.4 Looking Forward

Naive Bayes is one entry point into a vast landscape of probabilistic modeling. The ideas we've developed—priors, likelihoods, posterior inference, graphical models—extend far beyond classification:

- **Bayesian Networks** model complex systems with many interacting variables
- **Hidden Markov Models** extend these ideas to sequences, powering speech recognition and biological sequence analysis
- **Probabilistic graphical models** unify directed and undirected models for structured probabilistic reasoning
- **Bayesian deep learning** brings uncertainty quantification to neural networks

The foundation you've built—reasoning with probabilities, combining evidence, deciding under uncertainty—will serve you across all these areas and beyond.

The goal is not certainty. It's making the best decision with the information we have.

A Notation Reference

A.1 Core Symbols

Symbol	Meaning
m	Number of training examples
n	Number of features
K	Number of classes
$\mathbf{x}$	Feature vector: $\mathbf{x} = (x_1, x_2, \ldots, x_n)$
x_j	Value of feature j
Y	Class variable
c	A specific class (e.g., Bacterial, Viral)
$\hat{y}$	Predicted class

A.2 Probability Notation

Expression	Meaning
$P(Y = c)$	Prior probability of class c
$P(x_j \mid Y = c)$	Likelihood of feature j given class c
$P(Y = c \mid \mathbf{x})$	Posterior probability of class c given features $\mathbf{x}$
$p(x_j \mid Y = c)$	Probability density (for continuous features)
$P(\mathbf{x})$	Marginal probability of observing $\mathbf{x}$

A.3 Model Parameters

Symbol	Meaning
μ_{jc}	Mean of feature j in class c (Gaussian NB)
σ_{jc}	Standard deviation of feature j in class c
σ^2_{jc}	Variance of feature j in class c
α	Smoothing parameter (typically $\alpha = 1$)
d	Number of possible values for a categorical feature

A.4 Notational Conventions

- Capital letters (X, Y) denote random variables
- Lowercase letters (x, y, c) denote specific values
- Bold lowercase ($\mathbf{x}$) denotes vectors
- $P(\cdot)$ denotes probability mass (discrete)
- $p(\cdot)$ denotes probability density (continuous)
- log denotes natural logarithm (base e)
- arg max returns the argument maximizing the expression
- The symbol $\mid$ means "given" or "conditional on"