

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z3003 · DATA MINING: SCALING TO LARGE DATA

Classification II: Naive Bayes

Monday 2 November 2026 · 50 minutes

Dr. Innocent Nyalala | BS Semester 5 (DSAI) | IIT Madras Zanzibar

By the End of Today You Can

IIT Madras Zanzibar

1. Read a probability from a table of counts, including P(A | B)
2. Use Bayes' theorem, and explain the malaria-test surprise
3. Classify a new farm by hand with Naive Bayes
4. Fix zero counts with Laplace smoothing
5. Choose the variant: categorical, Gaussian, multinomial, Bernoulli, complement
6. Run it six ways (hand, Python, streaming, Spark, Weka) and explain its speed

Today's Plan (40 minutes + questions)

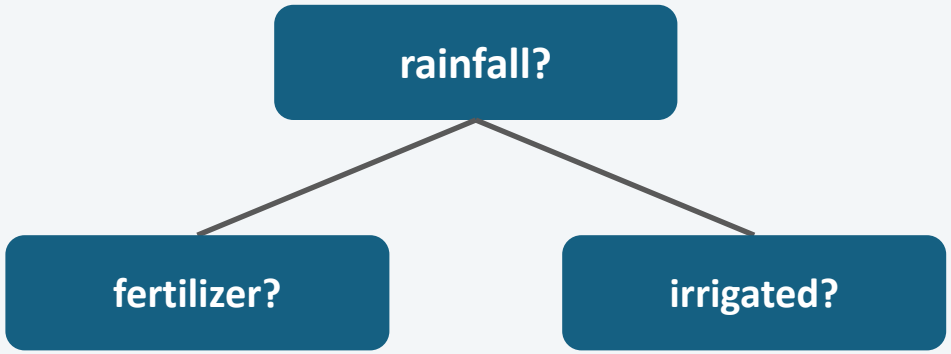
IIT Madras Zanzibar

minutes	part	what you get
0 – 4	Trees vs Naive Bayes	two different ways to decide
4 – 10	Probability from counts	P(High), P(High | fertilizer)
10 – 16	Bayes' theorem	turning a probability around; malaria test
16 – 25	Naive Bayes by hand	new farm; zero counts; smoothing; logs
25 – 32	The Naive Bayes family	Gaussian, multinomial, Bernoulli, complement
32 – 37	At scale + Weka demo	190,000 seasons; streaming; Spark; no-code Weka
37 – 40	Exercise and summary	then questions and further reading

Two Ways to Decide: Ask Questions, or Weigh Features

IIT Madras Zanzibar

Wednesday: Decision tree



asks questions one at a time,
follows one path

Today: Naive Bayes

rainfall Low	pushes → Low
fertilizer Yes	pushes → High
not irrigated	pushes → Low
smallholder	pushes → Low

weighs ALL features at once

Like a doctor combining fever, chills and headache: no single feature decides, together they do.

Our Data, and Probability Is Just Counting

IIT Madras Zanzibar

#	rainfall	size	fertilizer	irrigated	yield
1	Low	Large	No	Yes	Low
2	Low	Large	No	No	Low
3	High	Large	No	Yes	High
4	Medium	Medium	No	Yes	High
5	Medium	Small	Yes	Yes	High
6	Medium	Small	Yes	No	Low
7	High	Small	Yes	No	High
8	Low	Medium	No	Yes	Low
9	Low	Small	Yes	Yes	High
10	Medium	Medium	Yes	Yes	High
11	Low	Medium	Yes	No	High
12	High	Medium	No	No	High
13	High	Large	Yes	Yes	High
14	Medium	Medium	No	No	Low

Count the yield column:
9 rows say High, 5 rows say Low

P(High) = 9 ÷ 14 = 0.643
P(Low) = 5 ÷ 14 = 0.357

P(something) = the number of rows where it happens ÷ the number of rows we look at.
0.643 + 0.357 = 1.

The Bar '|' Means: Look Only Inside a Group

	High yield	Low yield	total
fertilizer Yes	6	1	7
fertilizer No	3	4	7
total	9	5	14

Each cell counts the farms (from the 14-row table) that have BOTH the row value and the column value.

Step 1 keep only the row 'fertilizer Yes': 7 farms

Step 2 inside that row, count High: 6 farms

Step 3 divide: 6 ÷ 7 = 0.857

P(High | fertilizer Yes) = 6 ÷ 7 = 0.857

P(A | B) = (number of farms that are A AND B at the same time) ÷ (number of farms that are B)
Here: (farms High AND fertilizer Yes = 6) ÷ (farms with fertilizer Yes = 7). 'AND' is one cell, never a sum.

Order Matters: P(A | B) Is Not P(B | A)

IIT Madras Zanzibar

	High yield	Low yield	total
fertilizer Yes	6	1	7
fertilizer No	3	4	7
total	9	5	14

P(fertilizer Yes | High)

look inside the High column: 9 farms

6 ÷ 9 = 0.667

P(High | fertilizer Yes)

look inside the fertilizer Yes row: 7 farms

6 ÷ 7 = 0.857

Same top (the 6 farms that are High AND fertilized), different bottom (9 vs 7).
Training data gives the left kind easily; to classify we need the right kind.

Bayes' Theorem: Four Parts, One Equation

IIT Madras Zanzibar

POSTERIOR		LIKELIHOOD		PRIOR		EVIDENCE
P(High | fertilizer)	=	P(fertilizer | High)	×	P(High)	÷	P(fertilizer)
what we want: chance of High once we know the farm used fertilizer		inside the 9 High farms, share that used fertilizer		share of all farms that are High, before knowing anything		share of all farms that used fertilizer
?		6 ÷ 9 = 0.667		9 ÷ 14 = 0.643		7 ÷ 14 = 0.5

	High yield	Low yield	total
fertilizer Yes	6	1	7
fertilizer No	3	4	7
total	9	5	14

In words:
posterior = likelihood × prior ÷ evidence
(likelihood: High column; prior and evidence: the totals)

Bayes' Theorem, Step by Step

step	what we do	working
1	likelihood × prior	0.667 × 0.643 = 0.429
2	÷ evidence	0.429 ÷ 0.5 = 0.857
3	**check by counting the table**	**6 ÷ 7 = 0.857 ✓**

	High yield	Low yield	total
fertilizer Yes	6	1	7
fertilizer No	3	4	7
total	9	5	14

likelihood 6 ÷ 9 = 0.667 (High column)
prior 9 ÷ 14 = 0.643 (total row)
evidence 7 ÷ 14 = 0.5 (total column)

If We Can Count, Why Do We Need Bayes?

IIT Madras Zanzibar

1	**One feature**	counting works; Bayes gives the same 0.857
2	**Many features**	no farm matches Low rain + Small + no fertilizer + no irrigation: 0 ÷ 0. With 20 yes/no features: 2^{20} > 1 million combinations
3	**Other direction**	labs measure P(positive | malaria); clinics need P(malaria | positive), with their own prior

Bayes is counting rearranged, so that we can use the counts we actually have.

The Malaria Test: Counting 10,000 People

IIT Madras Zanzibar

Given: 2% of people tested have malaria · test positive for 95% of sick people · test wrongly positive for 5% of healthy people

step	who	calculation	result
1	people with malaria	2% of 10,000 = 0.02 × 10,000	200
2	healthy people	10,000 − 200	9,800
3	sick AND test positive (true alarms)	95% of 200 = 0.95 × 200	190
4	healthy AND test positive (false alarms)	5% of 9,800 = 0.05 × 9,800	490
5	everyone whose test says positive	190 + 490	680
6	**share of positives who are really sick**	**190 ÷ 680**	**0.28 = 28%**

The Same Answer With Bayes' Theorem

IIT Madras Zanzibar

P(malaria | positive) = P(positive | malaria) × P(malaria) ÷ P(positive)

part	where it comes from	value
likelihood P(positive | malaria)	190 of the 200 sick	0.95
prior P(malaria)	200 of the 10,000 people	0.02
evidence P(positive)	680 of 10,000 (= 0.95 × 0.02 + 0.05 × 0.98)	0.068
posterior	**0.95 × 0.02 ÷ 0.068 = 0.019 ÷ 0.068**	**0.28**

Most people guess 95%. The truth is 28%: when a class is rare, most positives are false alarms.

Why 'Naive'? Use Each Feature Separately

IIT Madras Zanzibar

#	rainfall	size	fertilizer	irrigated	yield
1	Low	Large	No	Yes	Low
2	Low	Large	No	No	Low
3	High	Large	No	Yes	High
4	Medium	Medium	No	Yes	High
5	Medium	Small	Yes	Yes	High
6	Medium	Small	Yes	No	Low
7	High	Small	Yes	No	High
8	Low	Medium	No	Yes	Low
9	Low	Small	Yes	Yes	High
10	Medium	Medium	Yes	Yes	High
11	Low	Medium	Yes	No	High
12	High	Medium	No	No	High
13	High	Large	Yes	Yes	High
14	Medium	Medium	No	No	Low

The problem

no farm has Low rain AND Small AND no fertilizer AND no irrigation, so P(all four | High) cannot be counted

The naive assumption

inside one class, treat each feature separately

Then

P(all four | High) ≈ P(rain Low | High) × P(Small | High) × P(fert No | High) × P(irr No | High): four easy column counts

score(class) = P(class) × P(feature$_1$ | class) × P(feature$_2$ | class) × … × P(feature$_n$ | class). Bigger score wins.

Classify a New Farm, Step 1: Score for HIGH

IIT Madras Zanzibar

#	rainfall	size	fertilizer	irrigated	yield
1	Low	Large	No	Yes	Low
2	Low	Large	No	No	Low
3	High	Large	No	Yes	High
4	Medium	Medium	No	Yes	High
5	Medium	Small	Yes	Yes	High
6	Medium	Small	Yes	No	Low
7	High	Small	Yes	No	High
8	Low	Medium	No	Yes	Low
9	Low	Small	Yes	Yes	High
10	Medium	Medium	Yes	Yes	High
11	Low	Medium	Yes	No	High
12	High	Medium	No	No	High
13	High	Large	Yes	Yes	High
14	Medium	Medium	No	No	Low

shaded rows = the 9 High farms

New farm: rainfall Low · size Small · fertilizer No · irrigated No

feature	High farms with it	probability
prior	9 of 14 farms are High	9 ÷ 14 = 0.643
rainfall = Low	farms 9, 11	2 ÷ 9 = 0.222
size = Small	farms 5, 7, 9	3 ÷ 9 = 0.333
fertilizer = No	farms 3, 4, 12	3 ÷ 9 = 0.333
irrigated = No	farms 7, 11, 12	3 ÷ 9 = 0.333
score(HIGH)	**multiply all five**	**0.0053**

Step 2: Score for LOW

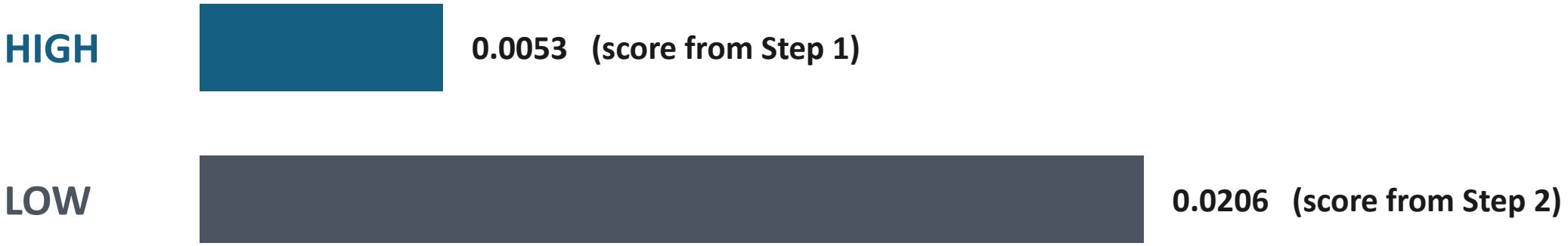
#	rainfall	size	fertilizer	irrigated	yield
1	Low	Large	No	Yes	Low
2	Low	Large	No	No	Low
3	High	Large	No	Yes	High
4	Medium	Medium	No	Yes	High
5	Medium	Small	Yes	Yes	High
6	Medium	Small	Yes	No	Low
7	High	Small	Yes	No	High
8	Low	Medium	No	Yes	Low
9	Low	Small	Yes	Yes	High
10	Medium	Medium	Yes	Yes	High
11	Low	Medium	Yes	No	High
12	High	Medium	No	No	High
13	High	Large	Yes	Yes	High
14	Medium	Medium	No	No	Low

shaded rows = the 5 Low farms

New farm: rainfall Low · size Small · fertilizer No · irrigated No

feature	Low farms with it	probability
prior	5 of 14 farms are Low	5 ÷ 14 = 0.357
rainfall = Low	farms 1, 2, 8	3 ÷ 5 = 0.6
size = Small	farm 6	1 ÷ 5 = 0.2
fertilizer = No	farms 1, 2, 8, 14	4 ÷ 5 = 0.8
irrigated = No	farms 2, 6, 14	3 ÷ 5 = 0.6
score(LOW)	**multiply all five**	**0.0206**

Step 3: Compare the Scores



	working	answer
total of both scores	0.0053 + 0.0206	0.0259
P(Low | new farm)	0.0206 ÷ 0.0259	0.80
P(High | new farm)	0.0053 ÷ 0.0259	0.20

Why divide by 0.0259? It makes the two scores add to 1. It is the evidence in Bayes' theorem.

Naive Bayes From Scratch: 11 Lines of Python

```
 1  import pandas as pd
 2  farms = pd.DataFrame(F, columns=["rain", "size", "fert", "irr", "yield"])
 3  new = {"rain": "Low", "size": "Small", "fert": "No", "irr": "No"}
 4  scores = {}
 5  for c in ["High", "Low"]:
 6      group = farms[farms["yield"] == c]          # look only inside class c
 7      score = len(group) / len(farms)             # prior P(class)
 8      for col, val in new.items():
 9          score *= (group[col] == val).mean()     # times P(feature | class)
10      scores[c] = round(score, 4)
11  print(scores)       # {"High": 0.0053, "Low": 0.0206}
```

line	what it does
2	the 14 farms as a table
3	the new farm's features
5	for each class
6	look only inside it (|)
7	start with the prior
8 – 9	multiply each P(feature | class)
11	scores: same as by hand

Training Naive Bayes is just this: counting shares inside each class.

The Zero-Count Trap

IIT Madras Zanzibar

Second new farm: rainfall High · size Small · fertilizer No · irrigated No

feature	HIGH (9 farms)	LOW (5 farms)
prior	9 ÷ 14 = 0.643	5 ÷ 14 = 0.357
rainfall = High	4 ÷ 9 = 0.444	0 ÷ 5 = 0 (no Low farm had High rain)
size = Small	3 ÷ 9 = 0.333	1 ÷ 5 = 0.2
fertilizer = No	3 ÷ 9 = 0.333	4 ÷ 5 = 0.8
irrigated = No	3 ÷ 9 = 0.333	3 ÷ 5 = 0.6
score	**0.0106**	**0.357 × 0 × 0.2 × 0.8 × 0.6 = 0**

Veto = one factor alone forces the score to 0 and overrules every other feature.
Without rainfall, LOW would win (0.034 vs 0.024). Zero in 5 farms means 'not seen yet', not 'impossible'.

The Fix: Laplace Smoothing (Add 1 to Every Count)

IIT Madras Zanzibar

P(value | class) = (count + 1) ÷ (class size + k) k = number of values the feature can take

Rainfall inside the 5 LOW farms (farms 1, 2, 6, 8, 14). Rainfall has 3 values, so k = 3.

rainfall value	count in LOW	count + 1	new share
Low	3	3 + 1 = 4	4 ÷ 8 = 0.5
Medium	2	2 + 1 = 3	3 ÷ 8 = 0.375
High	0	0 + 1 = 1	1 ÷ 8 = 0.125 (not 0)
total	**5 (class size)**	**5 + 3 = 8**	**1.0**

8 = 5 Low farms + 3 extra counts (one per rainfall value). Unseen → small, not zero: no veto.
scikit-learn: CategoricalNB(alpha=1); Weka NaiveBayes does this automatically.

Computers Add Logs Instead of Multiplying

IIT Madras Zanzibar

Problem: $0.1 \times 0.1 \times \ldots$ (400 times) $= 10^{-400}$, too small to store, so the computer gives 0.
Fix: $\ln(a \times b \times c) = \ln a + \ln b + \ln c$. Take ln of each number, then ADD down the column.

feature	HIGH: ln of each Step 1 number	LOW: ln of each Step 2 number
prior	ln 0.643 = −0.442	ln 0.357 = −1.030
rainfall Low	ln 0.222 = −1.504	ln 0.6 = −0.511
size Small	ln 0.333 = −1.099	ln 0.2 = −1.609
fertilizer No	ln 0.333 = −1.099	ln 0.8 = −0.223
irrigated No	ln 0.333 = −1.099	ln 0.6 = −0.511
add the column	**−5.242 ($e^{-5.242} = 0.0053$)**	**−3.884 ($e^{-3.884} = 0.0206$)**

Biggest sum wins (closest to zero): −3.884 > −5.242, so LOW, the same answer as multiplying.

The MAP Rule, and What Kind of Classifier This Is

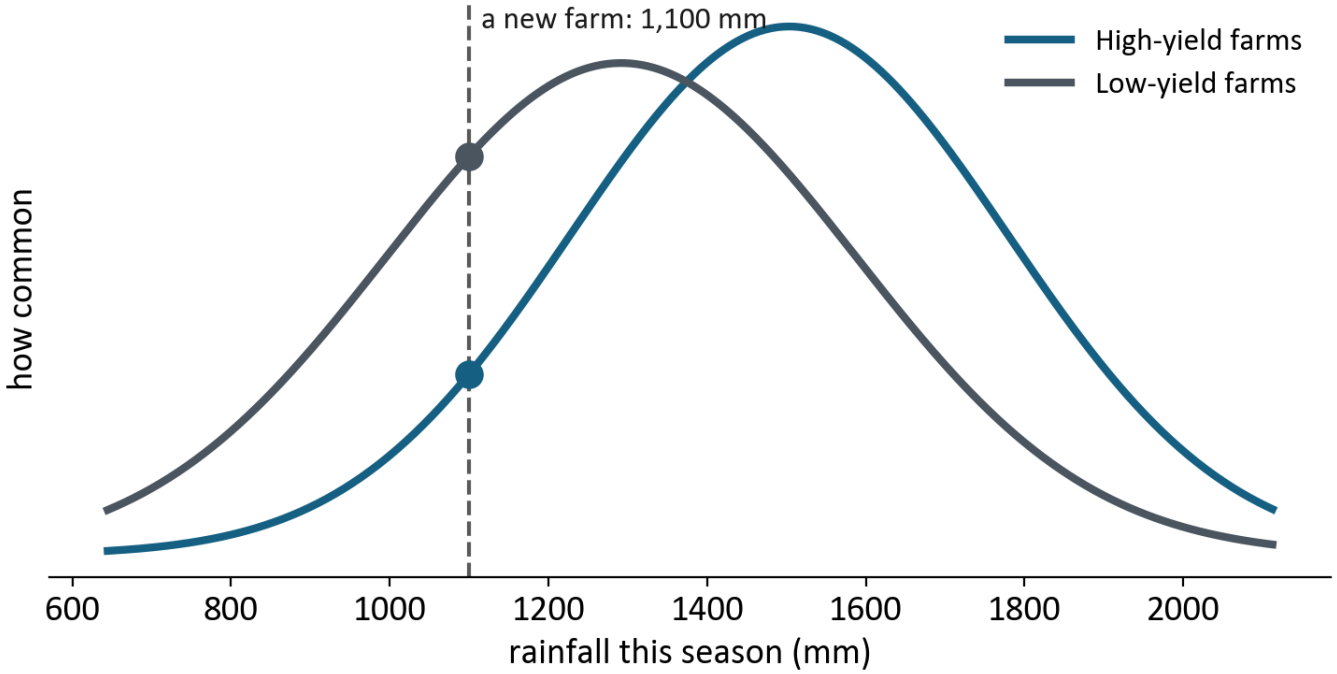
IIT Madras Zanzibar

MAP rule: predicted class = the class with the biggest P(class) × Π P(feature | class)

1	**MAP**	maximum a posteriori = choose the class with the biggest posterior
2	**Π (capital pi)**	multiply all of them together (like Σ means add all of them)
3	**For HIGH**	0.643 × 0.222 × 0.333 × 0.333 × 0.333 = 0.0053
4	**No evidence**	P(features) is the same for every class, so it cannot change the winner

Generative (Naive Bayes): learns what each class looks like, then asks which class explains the farm.
Discriminative (trees, logistic regression, SVM): learns only the border between classes.

Numbers Like Rainfall: Use a Bell Curve



Option 1	cut into bands and count (Week 3)
Option 2	one bell curve per class: average and spread
Read it	curve height at the farm's value

At 1,100 mm the Low curve is about twice as high, so this feature pushes towards LOW.

Gaussian Naive Bayes, Step by Step on Real Rainfall

IIT Madras Zanzibar

From the 190,000 seasons: HIGH average 1,503 mm (std 275), 49.5% of seasons · LOW average 1,292 mm (std 296), 50.5%

height(x) = 1 ÷ (std × √(2π)) × e^(−z² / 2), z = (x − average) ÷ std, √(2π) = 2.507, new farm x = 1,100 mm

step	HIGH seasons	LOW seasons
1 z	(1100 − 1503) ÷ 275 = −1.47	(1100 − 1292) ÷ 296 = −0.65
2 e^(−z²/2)	e^(−1.07) = 0.342	e^(−0.21) = 0.810
3 1 ÷ (std × 2.507)	1 ÷ 689 = 0.00145	1 ÷ 742 = 0.00135
4 height = 2 × 3	0.00050	0.00109
5 × prior	0.495 × 0.00050 = 0.000245	0.505 × 0.00109 = 0.000552
6 share of 0.000797	**0.31**	**0.69 → LOW**

Five Kinds of Naive Bayes: Pick by Data Type

IIT Madras Zanzibar

variant	data it expects	P(feature | class) comes from	typical use	scikit-learn
Categorical	categories	share of the class with that value	farm and survey records	CategoricalNB
Gaussian	numbers	bell curve: class mean and std	sensors, rainfall, lab tests	GaussianNB
Multinomial	counts	how often the word appears in the class	news, SMS, reviews	MultinomialNB
Bernoulli	yes / no	word present, and word absent	short messages, binary flags	BernoulliNB
Complement	counts, rare class	statistics of all OTHER classes	imbalanced text	ComplementNB

Same recipe for all five: prior × one probability per feature. Only the feature model changes.

Multinomial Naive Bayes: Classifying an SMS

IIT Madras Zanzibar

training message	class
free mbegu free	spam
free offer	spam
mbegu meeting	not spam
meeting today	not spam

word	count in spam	count in not spam
free	2 + 1 = 3	0
mbegu	1	1
offer	1	0
meeting	0	2
today	0	1
all words	5	4

New SMS: "free mbegu" vocabulary = 5 words

	spam	not spam
P(free | class)	(3 + 1) ÷ (5 + 5) = 0.4	(0 + 1) ÷ (4 + 5) = 0.111
P(mbegu | class)	(1 + 1) ÷ (5 + 5) = 0.2	(1 + 1) ÷ (4 + 5) = 0.222
prior	2 of 4 messages = 0.5	2 of 4 messages = 0.5
score	**0.5 × 0.4 × 0.2 = 0.040**	**0.5 × 0.111 × 0.222 = 0.012**

(3 + 1) ÷ (5 + 5): 3 = times 'free' appears in spam; + 1 = Laplace;
5 = all words in spam; + 5 = one extra count for each of the 5 vocabulary words.

Bernoulli vs Multinomial: Do Missing Words Count?

IIT Madras Zanzibar

Message: “free free mbegu”

	free	mbegu	offer	meeting	today
Multinomial (counts)	2	1	–	–	–
Bernoulli (present?)	yes	yes	no	no	no

Multinomial

absent words ignored; best for long documents

Bernoulli

absent words count as evidence; short texts, yes/no data

TF-IDF

down-weights common words; works with MultinomialNB

McCallum and Nigam (1998): multinomial is usually more accurate on longer documents.

Four Practical Problems and Their Fixes

IIT Madras Zanzibar

1. Choosing the smoothing

Laplace +1; Lidstone +α (e.g. 0.1); m-estimate. Choose α by cross-validation.

2. Missing values

leave the unknown feature out of the product; nothing else changes

3. Related features

the same evidence counted twice → over-confident; remove duplicates

4. Extreme probabilities

ranking right, percentages too close to 0 or 1; calibrate them

In scikit-learn: alpha= for smoothing; CalibratedClassifierCV for honest probabilities.

Naive Bayes in Python: 190,000 Farm-Seasons

```
 1  import pandas as pd
 2  from sklearn.naive_bayes import CategoricalNB
 3  from sklearn.model_selection import train_test_split
 4  d = pd.read_csv("zanzibar_farm_seasons.csv").dropna()
 5  d["rain_band"] = pd.qcut(d.rainfall_mm, 3, labels=["Low", "Med", "High"])
 6  features = ["rain_band", "fertilizer_used", "irrigated", "season", "farm_type"]
 7  X = d[features].apply(lambda c: c.astype("category").cat.codes)
 8  Xtr, Xte, ytr, yte = train_test_split(X, d.yield_class, test_size=0.3)
 9  nb = CategoricalNB(alpha=1).fit(Xtr, ytr)      # counting + smoothing
10  print(nb.score(Xte, yte))                       # about 0.70
```

line	what it does
1 – 3	load the tools
4	read the data
5	rainfall → 3 bands
6 – 7	the 5 features, as numbers
8	hide 30% as test
9	train = count
10	accuracy on hidden farms

Naive Bayes: 70.3% correct. Depth-6 tree on the same features: 70.7%. Training: under 0.5 seconds.

Does the Naive Guess Match Reality?

IIT Madras Zanzibar

Profile: rainfall Low · fertilizer Yes · not irrigated · Vuli season · smallholder

Naive Bayes says
P(High) = 30%
(multiplying 5 features)

Direct count says
34% High
(6,278 real farms with this profile)

Close, despite the naive assumption. Both put this farm on the LOW side.

Six Ways to Run Naive Bayes

IIT Madras Zanzibar

1. By hand

count tables (today)

2. From scratch

pandas groupby + value shares

3. scikit-learn

CategoricalNB, GaussianNB, MultinomialNB, BernoulliNB, ComplementNB

4. Streaming

partial_fit(chunk): data never fully in memory

5. Distributed

Spark NaiveBayes: count on many machines, add

6. No code

Weka NaiveBayes; R: e1071, naivebayes

Underneath, all six do the same three things: count, smooth, multiply.

Naive Bayes on Data Too Big for Memory

IIT Madras Zanzibar

```python
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
cols = ["fertilizer_used", "irrigated"]
nb = BernoulliNB()
for chunk in pd.read_csv("seasons.csv", chunksize=50_000):
    X = (chunk[cols] == "Yes").astype(int)
    nb.partial_fit(X, chunk.yield_class,
                   classes=["High", "Low"])
print(nb.class_count_)        # counts summed over all chunks
```

```python
# PySpark: counting over a cluster
from pyspark.ml.classification import (
    NaiveBayes)
nb = NaiveBayes(modelType="multinomial",
                smoothing=1.0)
model = nb.fit(train_df)      # one pass
pred = model.transform(test_df)
```

machine 1: counts | **machine 2: counts** | **machine 3: counts** → **add the counts → model**

> **Counts add up, so chunks, streams and clusters all give the same model as one big pass.**

Naive Bayes in Weka: No Code (5-Minute Demo)

IIT Madras Zanzibar

1	**Open**	Weka GUI Chooser → Explorer → Preprocess → Open file → farms14.arff
2	**Choose**	Classify tab → Choose → bayes → NaiveBayes
3	**Train**	Test options: Use training set → Start → read the count tables
4	**Predict**	Supplied test set: new_farm.arff; More options → Output predictions → Low

Weka adds 1 to every count automatically (Laplace).
For big data: open farm_seasons_5000.csv, use 10-fold cross-validation, compare NaiveBayes with J48.

Naive Bayes Draws a Straight Border

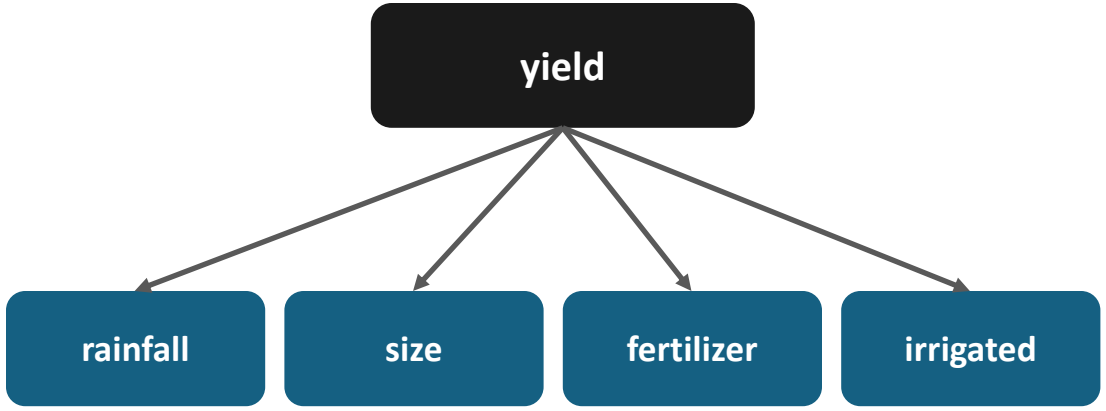
IIT Madras Zanzibar

$$\log \text{score(spam)} - \log \text{score(not spam)} = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots \rightarrow \text{a straight border}$$

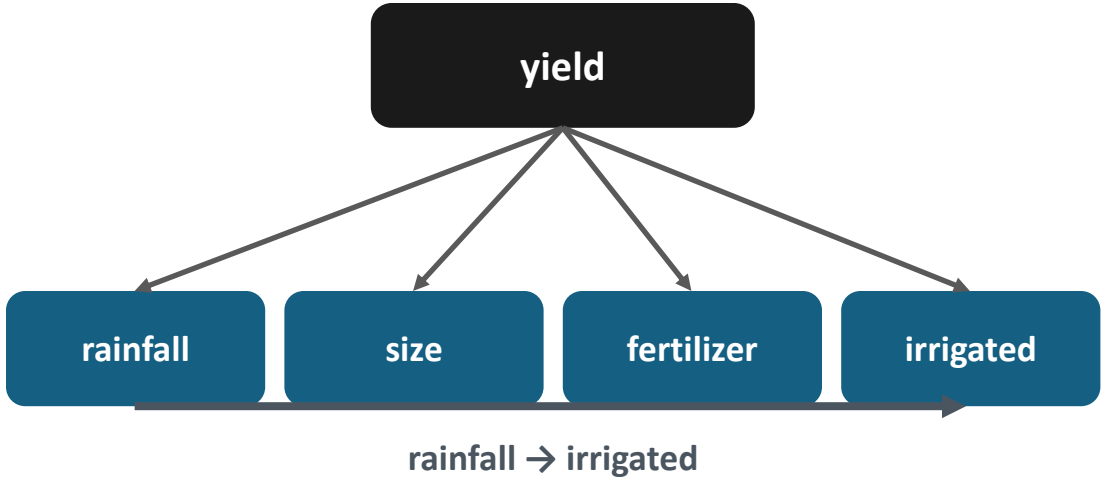
	Naive Bayes	Logistic regression
how weights are learned	counting, one pass	iterative search
little data	reaches its best quickly	needs more examples
lots of data	good	usually more accurate
type	generative	discriminative

> **Ng and Jordan (2002): Naive Bayes wins with small data, logistic regression with large data.**

Beyond Naive: Letting Features Talk to Each Other



Naive Bayes: no arrows between features



TAN: each feature may get one extra parent

Semi-naive Bayes: TAN (Friedman et al., 1997), AODE (Webb et al., 2005).
In Weka: BayesNet (TAN search), AODE package.

Why Naive Bayes Is a Data Mining Favorite

IIT Madras Zanzibar

One pass

training = counting; no search for splits

Parallel and streaming

count on many machines, add up; update as data arrives

Little data

few numbers to estimate

Weakness

related features get counted twice: over-confident

A strong, fast first try, especially for text and huge or streaming data.

Naive Bayes in the Real World

IIT Madras Zanzibar

Spam filters

each word is a feature

SMS and news

topic or language

Health

combine several symptoms

Reviews

positive or negative?

Mobile money

fast fraud risk score

Agriculture

pest risk from weather features

Common thread: many weak features, combined quickly into one decision.

Your Turn (3 minutes, pairs)

IIT Madras Zanzibar

10 SMS	spam (4)	not spam (6)
contains 'free'	3	1
contains 'win'	2	0

1. P(spam), P(free | spam), P(free | not)
2. Message has 'free': P(spam | free)?
3. Message has 'free' and 'win': compare scores
4. Redo 3 with Laplace (+1, k = 2)

Recipe: score(class) = P(class) × P(word$_1$ | class) × P(word$_2$ | class). Bigger score wins.

Solution 1 and 2: Prior, Likelihoods, Bayes

IIT Madras Zanzibar

quantity	how	working
P(spam)	4 of 10 messages	4/10 = 0.4
P(free | spam)	3 of the 4 spam	3/4 = 0.75
P(free | not)	1 of the 6 normal	1/6 = 0.167
top	P(free | spam) × P(spam)	0.75 × 0.4 = 0.30
bottom: P(free)	4 of 10 (= 0.30 + 0.167 × 0.6)	0.40
P(spam | free)	**top ÷ bottom**	**0.30 ÷ 0.40 = 0.75**

Solution 3: 'free' and 'win', Without Smoothing

IIT Madras Zanzibar

class	prior	P(free | class)	P(win | class)	score
spam	0.4	3/4 = 0.75	2/4 = 0.5	0.4 × 0.75 × 0.5 = 0.15
not spam	0.6	1/6 = 0.167	0/6 = 0	0.6 × 0.167 × 0 = 0

Spam wins, but only because a single zero vetoed the other class: the zero-count trap.

Solution 4: With Laplace Smoothing

IIT Madras Zanzibar

class	P(free | class)	P(win | class)	score
spam	(3+1)/(4+2) = 0.667	(2+1)/(4+2) = 0.5	0.4 × 0.667 × 0.5 = 0.133
not spam	(1+1)/(6+2) = 0.25	(0+1)/(6+2) = 0.125	0.6 × 0.25 × 0.125 = 0.019

Spam wins 0.133 vs 0.019, now on the evidence, with no zero veto.

Summary

IIT Madras Zanzibar

1	**Probability**	count ÷ total; '|' = look only inside a group
2	**Bayes' theorem**	posterior = likelihood × prior ÷ evidence; rare class → false alarms
3	**Naive Bayes**	prior × one probability per feature; bigger score wins
4	**Smoothing and speed**	add 1 to avoid zero vetoes; training = counting

Next: overfitting and pruning, why a tree that asks too many questions fails on new data.

Questions (5 minutes)

IIT Madras Zanzibar

Q: If 'independent' is false, why use it?

A: We only need the right class to score highest, and that usually survives.

Q: Naive Bayes or a decision tree?

A: NB: fast, many features (text). Tree: readable rules, related features. Try both.

Q: Why do programs add logs instead of multiplying?

A: Tiny products underflow; adding logarithms gives the same ranking.

Q: Do I need to normalize the data?

A: No for counts. GaussianNB uses each class's own mean and spread.

Further Reading: Books and Free Resources

IIT Madras Zanzibar

resource	why read it	access
Han, Pei & Tong, Data Mining: Concepts and Techniques, 4th ed. (2022)	course text; Bayes classification	library
Tan, Steinbach, Karpatne & Kumar, Introduction to Data Mining, 2nd ed. (2019)	clearest worked examples	library
Witten, Frank, Hall & Pal, Data Mining: Practical ML Tools and Techniques, 4th ed. (2017)	the Weka book	library
Leskovec, Rajaraman & Ullman, Mining of Massive Datasets, 3rd ed.	large-scale counting, MapReduce	free: mmds.org
Jurafsky & Martin, Speech and Language Processing (3rd ed. draft)	Naive Bayes for text and sentiment	free: web.stanford.edu/~jurafsky/slp3
scikit-learn User Guide: Naive Bayes	all five variants, with code	free: scikit-learn.org
Weka	free no-code workbench	free: ml.cms.waikato.ac.nz/weka

Further Reading: Key Research Papers

IIT Madras Zanzibar

paper	what it shows
Domingos & Pazzani (1997), Machine Learning 29: 103–130	why NB works even when features are dependent
McCallum & Nigam (1998), AAAI Workshop on Learning for Text Categorization	multinomial vs Bernoulli for text
Ng & Jordan (2002), NIPS 14	Naive Bayes vs logistic regression
Rennie, Shih, Teevan & Karger (2003), ICML	Complement Naive Bayes
Friedman, Geiger & Goldszmidt (1997), Machine Learning 29: 131–163	TAN: tree-augmented Naive Bayes
Webb, Boughton & Wang (2005), Machine Learning 58: 5–24	AODE: averaged one-dependence estimators

Tutorial 6 and the Weka files (farms14.arff, new_farm.arff, farm_seasons_5000.csv) are in Week_6/weka_demo.