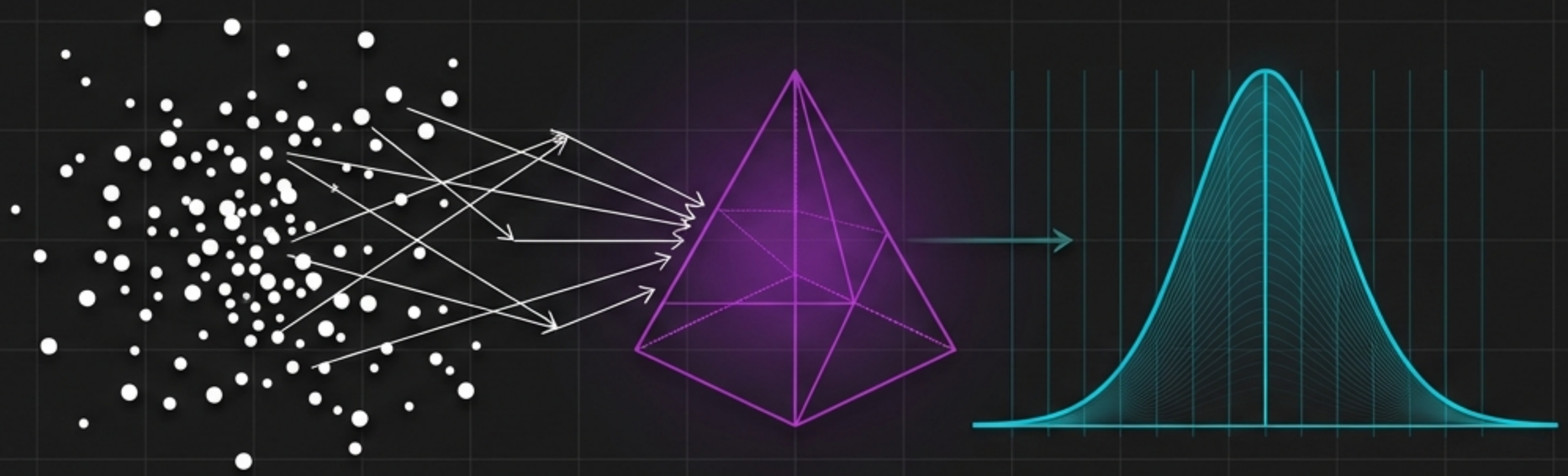
Naive Bayes Classifier

From Probability to Prediction

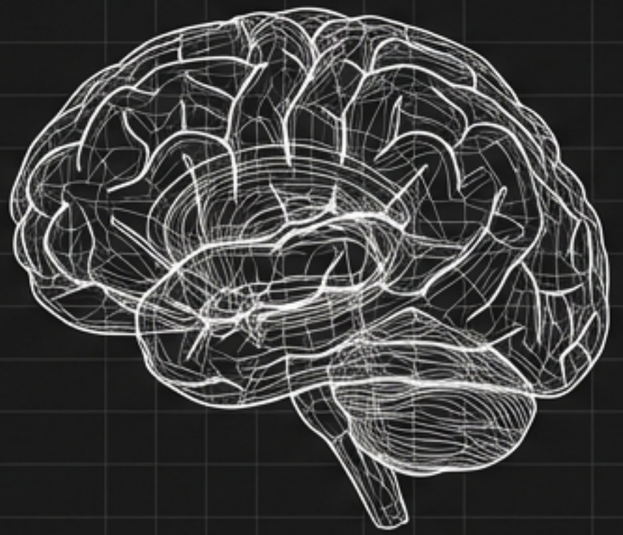
A visual guide to theory, algorithm variants, and Python implementation.



Transformation of raw data into structured probability.

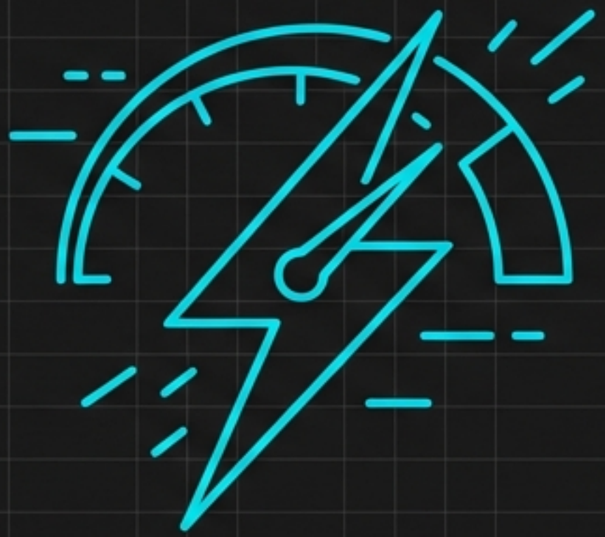
Self-Paced Learning Module

THE CORE CONCEPT



COMPLEXITY

Despite its "naive" assumption, the algorithm excels in complex real-world tasks like spam filtering, sentiment analysis, and medical diagnosis.

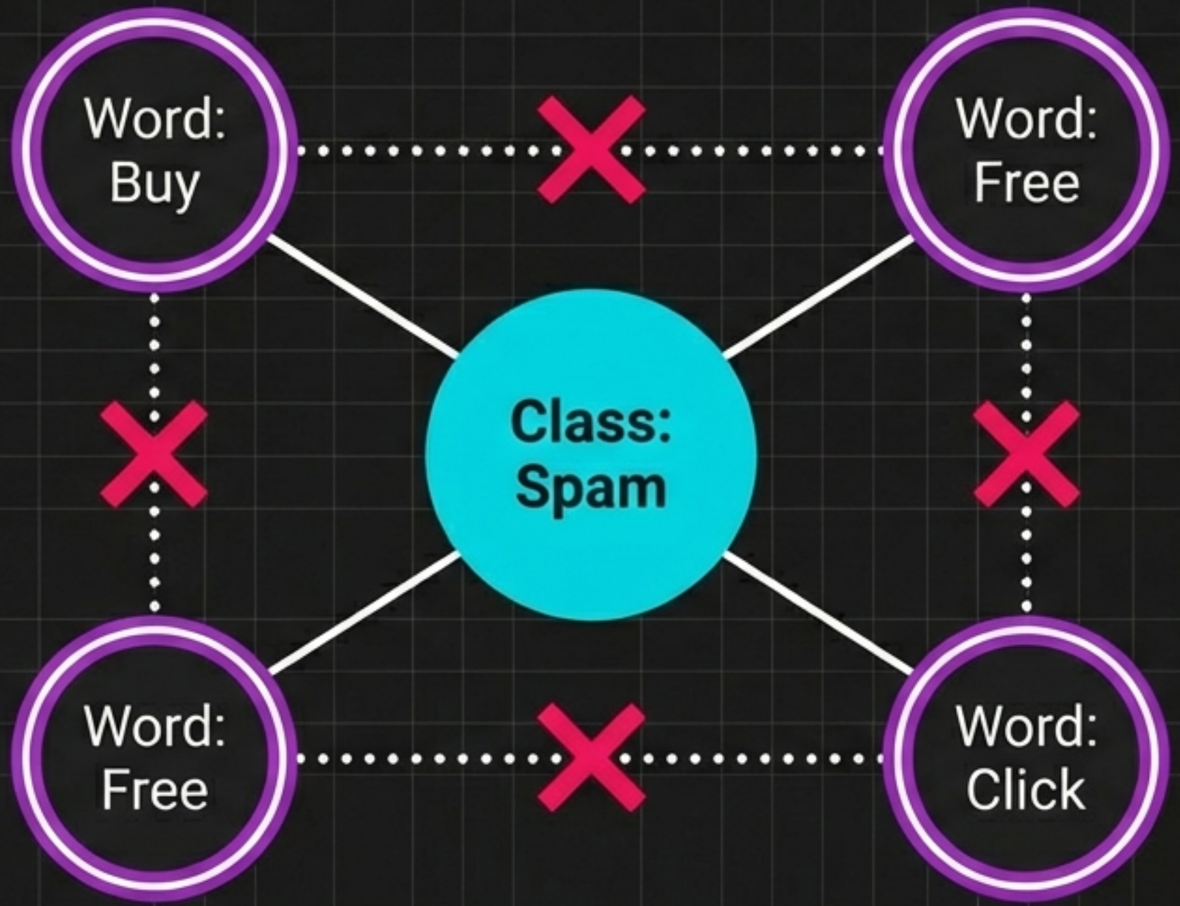


EFFICIENCY

Naive Bayes is a probabilistic classifier known for extreme speed. It calculates the probability of a data point belonging to a class and assigns it the highest value.

Why is it called “Naive”?

The algorithm assumes that features are INDEPENDENT of each other given the class.



Example: The presence of ‘Buy’ does not affect the probability of ‘Free’. They are treated as separate pieces of evidence.

The Math: Bayes' Theorem

Posterior: Probability of Class given Data (Target)

Likelihood: Probability of Data given Class

$$P(A|B) = \frac{[P(B|A) * P(A)]}{P(B)}$$

Posterior: Probability of Class given Data (Target)

Evidence

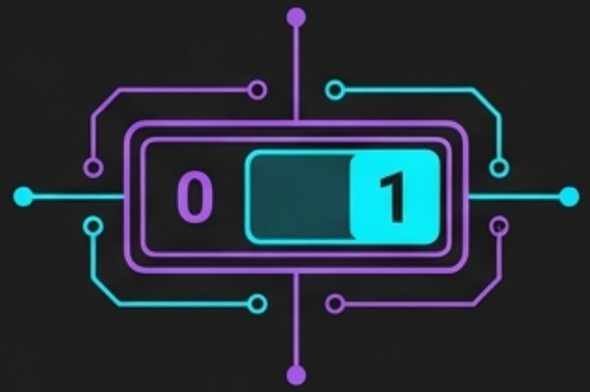
Prior: Initial Probability of Class

```
// Optimization Note:
Since the denominator P(B) is the same for all classes, we ignore it.
We simply calculate the numerator and pick the highest value.
```

One Algorithm, Three Variants

The logic remains the same, but the method adapts to the data shape.

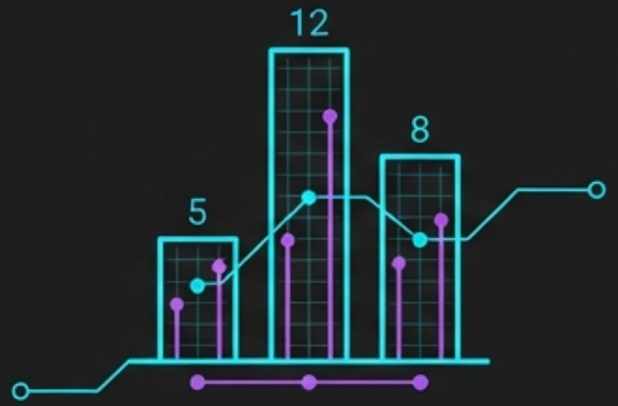
Bernoulli NB



For Binary/Boolean Data (0/1)

```
from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB() # data: [0, 1, 0, 1]
```

Multinomial NB



For Count Data (Word Frequencies)

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB() # data: [3, 0, 1, 5]
```

Gaussian NB

For Continuous Data (Decimals/Measurements)

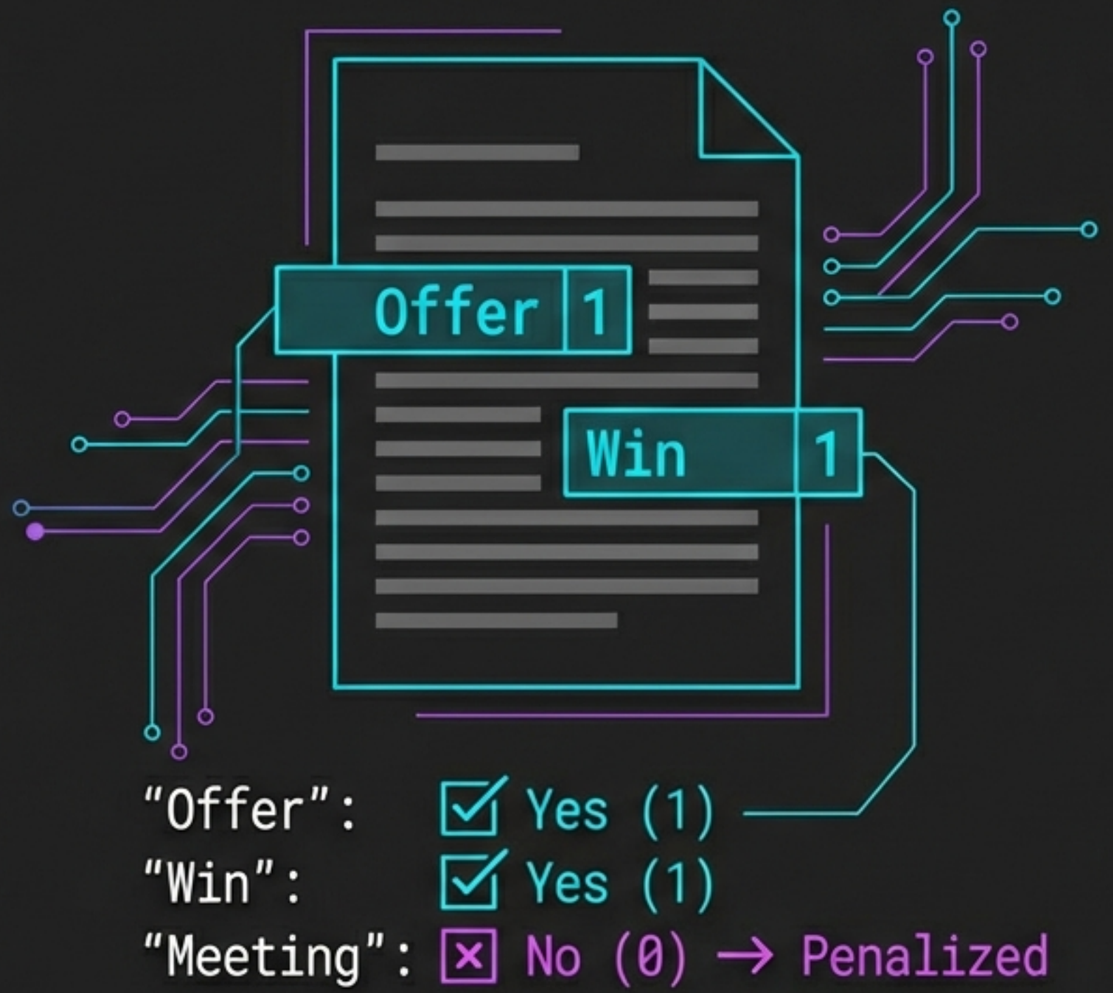
```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB() # data: [1.2, 3.5, 0.8, 2.1]
```

1. Bernoulli Naive Bayes

Data Type: Binary / Boolean (0 or 1).

Mechanism: Models the **PRESENCE** or **ABSENCE** of a feature.

Key Feature: Explicitly penalizes the **non-occurrence** of a feature (if a word is missing, it lowers the score).

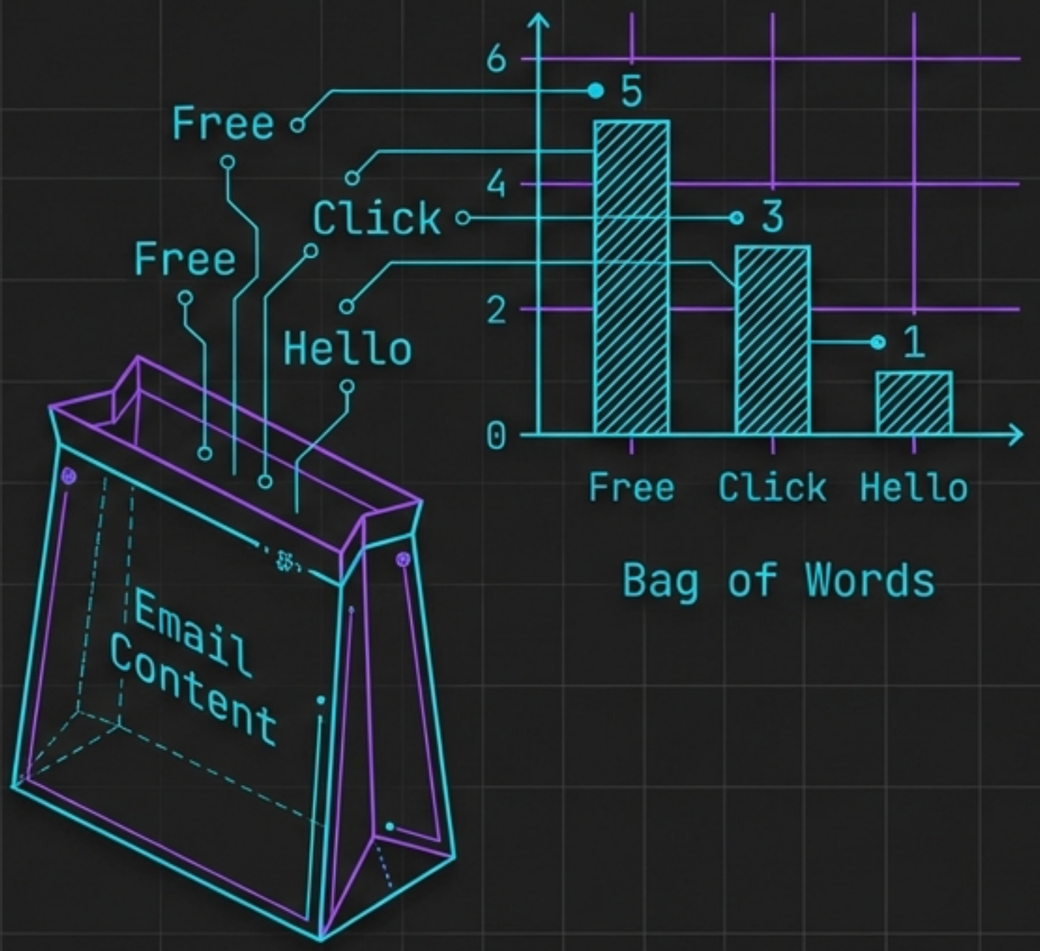


2. Multinomial Naive Bayes

Data Type: Discrete Counts.

Mechanism: Uses feature frequencies.

Use Case: The standard for Text Classification (NLP).

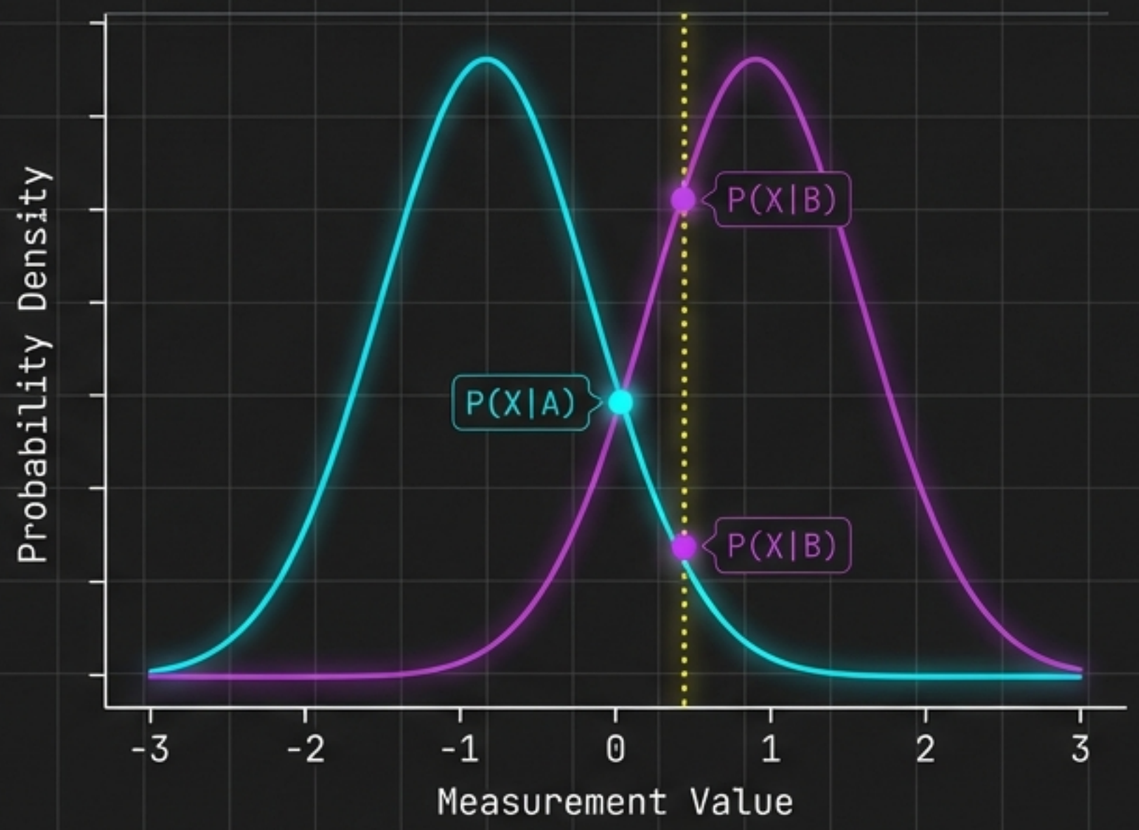


3. Gaussian Naive Bayes

Data Type: Continuous Numerical Values (e.g., Height, Weight, Sensors).

Assumption: Values follow a Normal Distribution (Bell Curve).

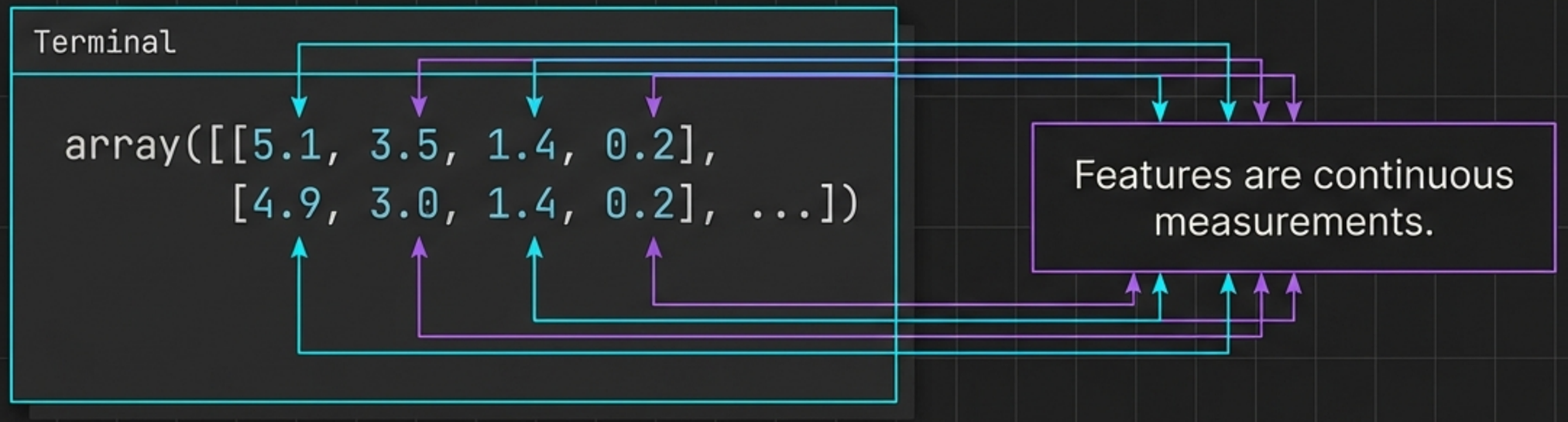
Mechanism: Uses Mean and Variance to estimate probability density.

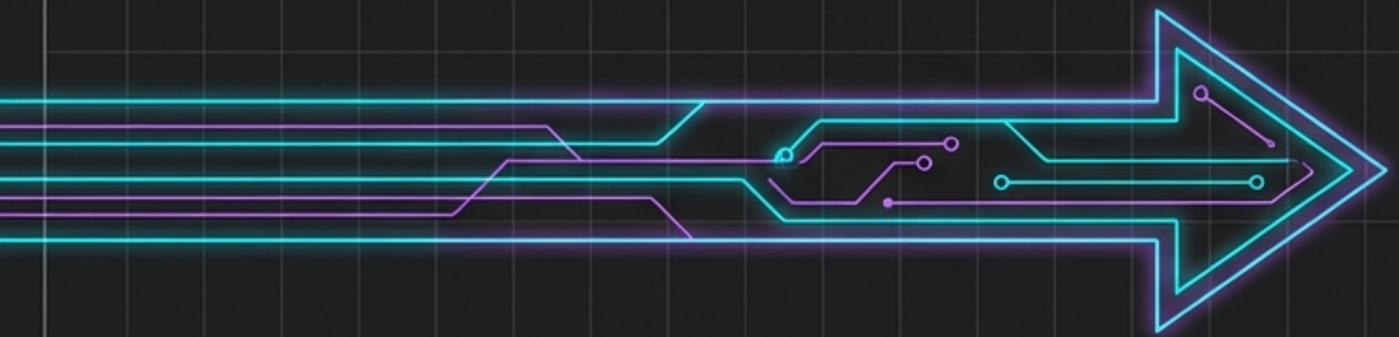


Class A: Cyan Curve (Mean < 0)
Class B: Purple Curve (Mean > 0)

The Scenario: Iris Flower Classification

Why we choose Gaussian NB



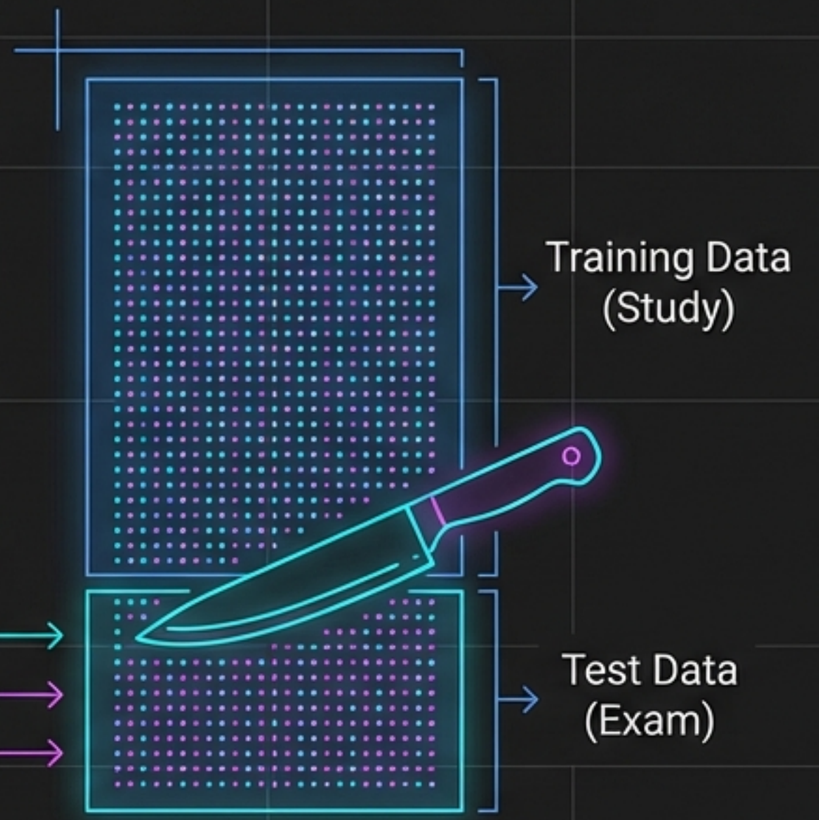


Therefore: Gaussian Naive Bayes

Step 1: Import & Split

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0
)
```



Step 2: Training the Model

Terminal

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

Under the Hood

Calculating Mean & Variance...

Step 3: Making Predictions

```
y_pred = gnb.predict(X_test)
print(y_pred)
```

Output Visualization

```
array([1,  0,  2,  1,  1,  0,  1,  2,  1...])
```

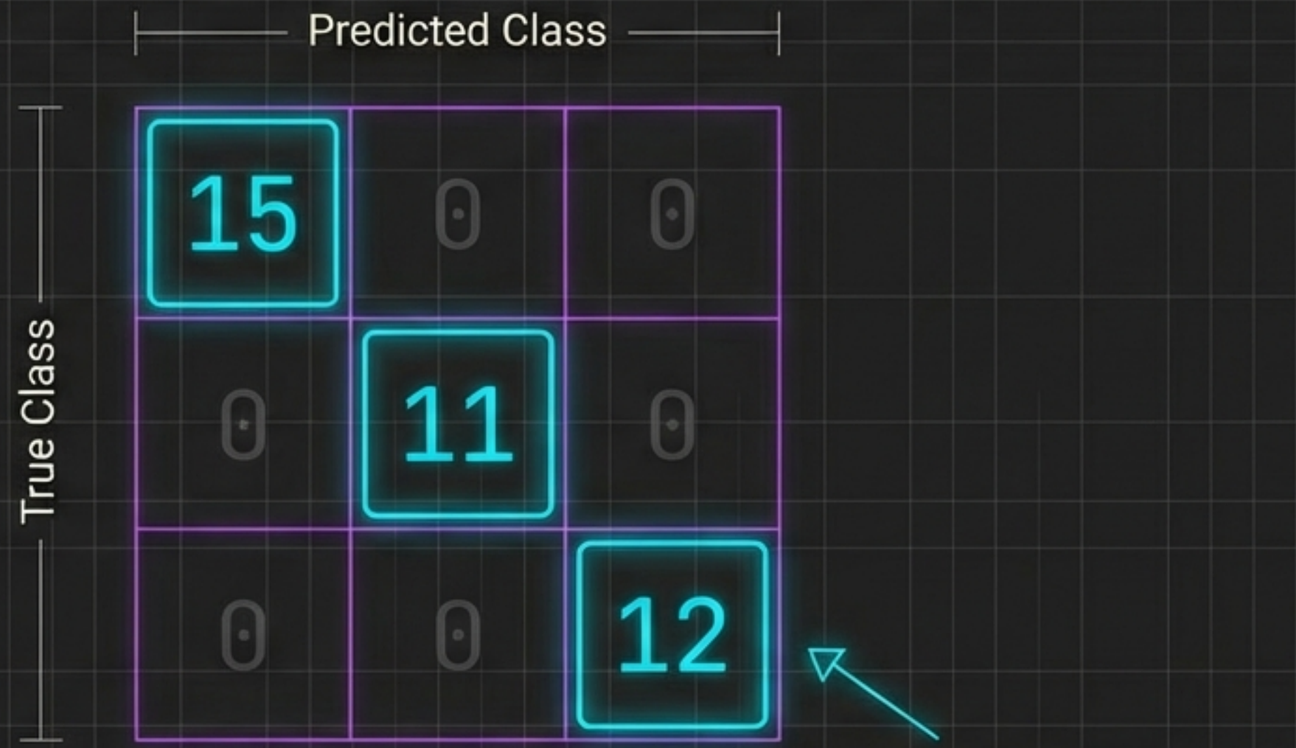
Roboto regular font
The model assigns a class label (0, 1, 2) to each unseen test flower.

Step 4: Evaluation

1.0 (100%)

Terminal

```
print(confusion_matrix(
    y_test, y_pred))
```



Summary Checklist

The Concept: **Naive Bayes** uses **probability** + **independence assumption** for speed.

The Selection: **Bernoulli** (**Binary**), **Multinomial** (**Counts**), **Gaussian** (**Continuous**).

The Result: **Scikit-Learn implementation** requires **minimal tuning** for high accuracy.

Simple assumptions. Powerful predictions.

Naive Bayes demonstrates that complex problems don’t always require complex deep learning.