DATA 2060: Gaussian Naive Bayes for classification

Zhiwei He, Lin Zhou, Linzhuo Zhang, Xiaoqing Yao

Outline

The Mathematics of GNB

Delve into the core probabilistic principles and essential equations.

Implementation & Pseudocode

Explore the numerical methods and algorithmic steps involved.

Sklearn Comparison

Validating our custom GNB with scikit-learn's results.

Key Insights & Challenges

Summarize interesting aspects and implementation hurdles.

Naive Bayes Overview

Naive Bayes (NB) is a probabilistic classification algorithm rooted in Bayes' theorem. It relies on a strong assumption: features are conditionally independent given the class label.

Probabilistic Nature

Calculates the probability of a data point belonging to a specific class.

Independence Assumption

Assumes that the presence of one feature does not affect the others.

Efficiency

Requires no iterative optimization, making training extremely fast.

The Mathematics of Naive Bayes

1. Training Phase

The model learns class priors and feature likelihoods. We use Laplace smoothing (+1) to prevent zero probabilities.

$$P(x_j = a \mid y = k) = \frac{\text{count}(x_j = a, y = k) + 1}{\text{count}(y = k) + m}$$

2. Prediction Phase

For a new sample, the model computes the posterior probability and selects the class with the maximum value.

$$\hat{y} = \arg\max_k [P(y = k) \prod_j P(x_j \mid y = k)]$$

Evaluating Standard Naive Bayes

1

2

Advantages

Fast and interpretable. Training only requires counting frequencies, with no complex gradient descent. It performs surprisingly well even with limited training data.

Disadvantages

Standard NB assumes categorical inputs. Strong feature correlations or mismatched distributions can degrade accuracy. It struggles with continuous variables without modification.

Solution: Gaussian Naive Bayes (GNB)

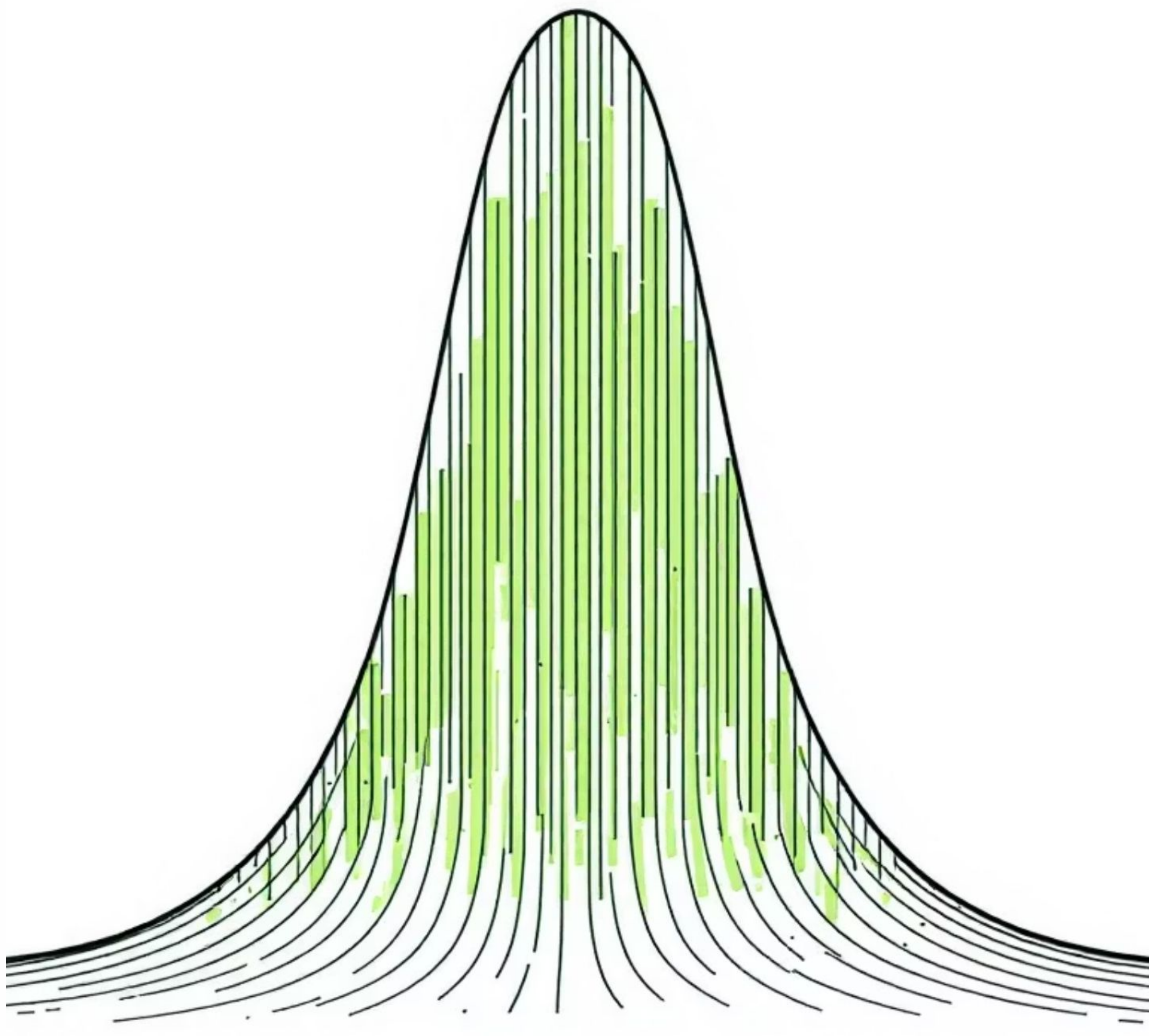
To handle continuous data, GNB assumes features follow a Normal (Gaussian) distribution rather than counting discrete frequencies.

Modeling Probability Densities

Instead of bins, GNB models data using mean and variance. This improves generalization for continuous variables like temperature or blood pressure.

The Gaussian Formula:

$$P(x_j \mid y = k) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$



Advantages of GNB

Handles continuous data

Models continuous features directly with a normal distribution, preventing information loss from discretization.

Fast and simple

Training is highly efficient, requiring only mean and variance estimation without iterative optimization.

Works with small datasets

Requires estimating few parameters, making it effective even with limited training data.

Limitations of GNB

→ **Assumes Gaussian distribution**

If features deviate significantly from a normal distribution, probability estimates can be inaccurate.

→ **Still assumes feature independence**

Strong correlations between features violate this assumption, potentially reducing prediction accuracy.

→ **Sensitive to outliers**

Outliers can skew mean and variance estimates, leading to unstable and unreliable predictions.

Representation

Each feature (x_j) for class (y = k) is modeled as:

$$P(x_j \mid y = k) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} \exp\left(-\frac{(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

During prediction, GNB computes:

$$P(y = k \mid x) \propto P(y = k) \prod_{j=1}^{d} P(x_j \mid y = k)$$

and predicts:

$$\hat{y} = \text{argmax}_k \, P(y = k) \prod_{j} P(x_j \mid y = k)$$

Loss Function

GNB implicitly minimizes the Negative Log-Likelihood (NLL) to encourage high probability for true class labels:

$$\mathrm{L}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \log P(y_i \mid \mathbf{x}_i; \theta)$$

Optimizer

GNB solves the optimization problem:

$$\theta = \arg\min_{\theta} L(\theta) = \arg\min_{\theta} [-\frac{1}{N} \sum_{n=1}^{N} \log P(y_i \mid \mathbf{x}_i; \theta)]$$

Where θ represents the Gaussian parameters for each class and feature:

$$\theta = \{\mu_{jk}, \sigma_{jk}^2 \mid j = 1, \dots, d; k = 1, \dots, K\}$$

GNB solves this in closed form using Maximum Likelihood Estimation (MLE):

$$\hat{\mu}_{jk} = \frac{1}{N_k} \sum_{\mathbf{x}_i \in \text{class } k} x_{ij}$$

$$\hat{\sigma}_{jk}^2 = \frac{1}{N_k} \sum_{\mathbf{x}_i \in \text{class } k} (x_{ij} - \hat{\mu}_{jk})^2$$

$$\hat{P}(y = k) = \frac{N_k}{N}$$

Pseudocode: Training Phase

```
Inputs:
  Training data: features X ∈ ℝnxd
  Labels: y ∈ {1, …, K}

For each class k ∈ {1, …, K}:
  Collect all samples belonging to class k:
    I□ = { i | yᵢ = k }
  Compute:
    Prior probability: P(y = k) = |I□| / n
    Mean for each feature j: μ□□ = mean(X[I□, j])
    Variance for each feature j: σ□□² = var(X[I□, j])
```

Pseudocode: Prediction Phase

```
For a new sample x = (x₁, x₂, …, x_d):
  For each class k ∈ {1, …, K}, compute:
    Likelihood (Gaussian assumption):
      P(x | y = k) = ∏□ [ 1 / √(2πσ□□²) * exp( −(x□ − μ□□)² / (2σ□□²) ) ]
    Combine with class prior:
      P(y = k | x) ∝ P(y = k) × P(x | y = k)
  Predict the class with highest posterior:
    ŷ = argmax□ P(y = k | x)
```

Dataset Application: Diabetes

We applied the model to the Kaggle Diabetes Dataset to predict patient outcomes based on medical metrics.

768

Patient Records

Total samples in the dataset.

9

Features

Continuous variables including Glucose, BMI, and Insulin.

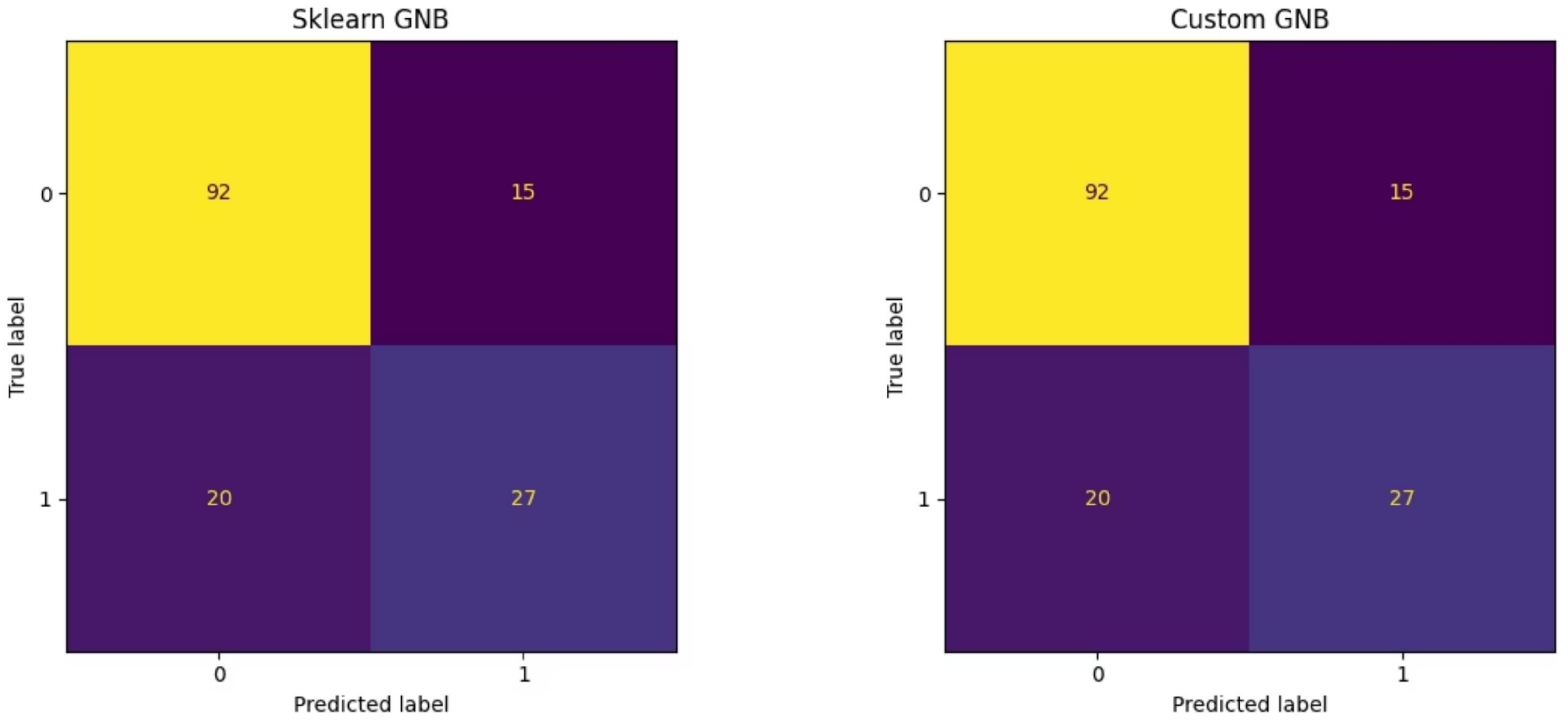
0-1

Scaling

Min-Max preprocessing applied to normalize data.

Model Comparison

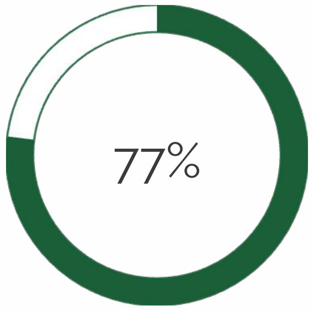
We compared our custom implementation against the standard skLearn GaussianNB. The confusion matrices below confirm that both models produced identical predictions.



Performance Results

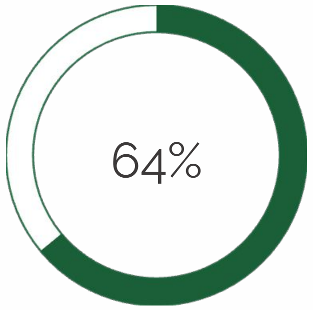
Both the Custom GNB and Sklearn GaussianNB implementations achieved identical performance metrics, validating the accuracy and reliability of our custom solution.

Custom GNB Implementation



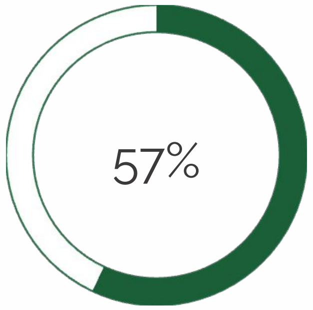
Accuracy

Correct predictions



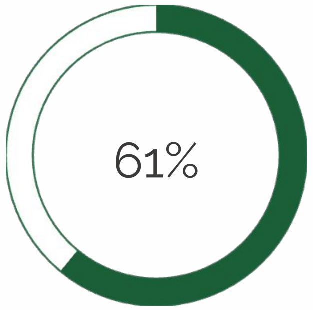
Precision

Positive predictions



Recall

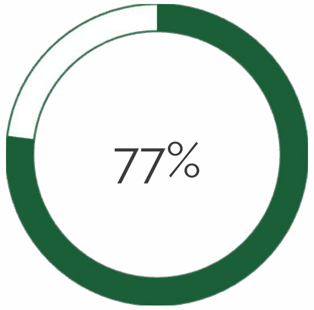
Positive case sensitivity



F1 Score

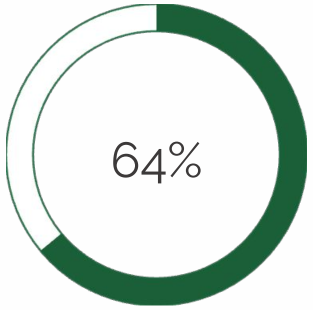
Precision & recall balance

Sklearn GaussianNB



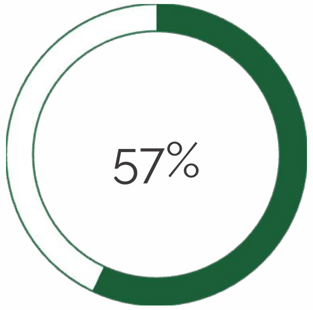
Accuracy

Correct predictions



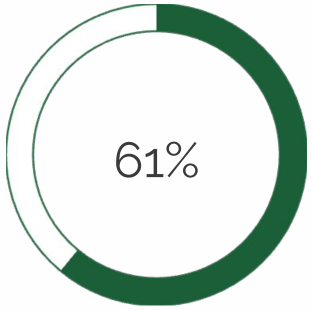
Precision

Positive predictions



Recall

Positive case sensitivity



F1 Score

Precision & recall balance

Summary

What we found interesting:

- GNB uses direct parameter calculation, making training fast and efficient without complex optimization.
- Despite its simple assumptions, GNB offers robust performance, especially in high-dimensional datasets.
- Our custom GNB effectively classified medical data, proving its utility in healthcare analytics.

Challenges:

- Numerical stability - handling log-likelihood calculations to prevent underflow with very small probabilities.
- Zero variance edge cases - dealing with features that have zero variance to avoid division by zero.
- Feature scaling - ensuring proper normalization for continuous variables.

Q&A

Thank you for your attention! We're now open for any questions you might have about our Gaussian Naive Bayes model.