

# Implementing Gaussian Naive Bayes to Coronary Heart Disease

Danielle Whisnant, Roshan Parikh, Wali Siddiqui, Rio Wombacher

# Overview

- Gaussian Naive Bayes (GNB) is a generative model
  - **Assume** that the data for each feature is conditionally independent given the class label and features follow Gaussian (normal) distributions.
- Unlike standard Naive Bayes, GNB specifically models continuous features
- For a given input, the probabilities are calculated for each class, and the final classification is assigned to the class with the highest posterior probability.
- Formal equation:

$$P_{\theta}(\mathbf{x}, y) = P_{\theta}(y) \prod_{i=1}^d P_{\theta}(x_i | y)$$

# Model Parameters

- Gaussian Naive Bayes, being a generative model, does not use an optimizer function.
- Instead, it capitalizes on the assumptions and uses closed-form Maximum Likelihood Estimation (MLE) to estimate parameters.
- MLE determines the parameters  $\mu_y$ ,  $\sigma^2_y$ ,  $P(y)$  that maximize the likelihood of the observed data. This is equivalent to minimizing the log loss.
  - Class Priors  $P(y)$ : the proportion of observations in each class.
  - Feature Means  $\mu_y$ : the mean of each feature  $x_i$  given class  $y$
  - Feature Variances  $\sigma^2_y$ : the variance of each feature  $x_i$  given class  $y$ .
- MLE formal equation:

$$\arg \min_{\theta} \sum_{i=1}^m -\log [P_{\theta}(x_i, y_i)]$$

# Making Predictions

- Utilizing our assumption, we calculate the predicted probabilities for each class  $y$  using the conditional probabilities for each feature  $x_i$ , assuming normal distributions
  - Note: Unlike other Naive Bayes classifiers, GNB does not use Laplace smoothing because it works with continuous features. Instead, variance smoothing is applied by adding a very small constant (e.g.,  $10^{-6}$ ) to the variance to avoid instability when variance is zero.
- We then convert our equation to logspace to avoid underflow or overflow

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$
$$\Rightarrow \log P(x_i, y) = -\frac{1}{2} \log(2\pi\sigma_y^2) - \frac{(x_i - \mu_y)^2}{2\sigma_y^2}$$

# Prediction steps

1. **Compute Conditional Probabilities:** For each feature  $x_i$  and class  $y$ , calculate  $P(x_i|y)$  using the Gaussian probability density function.
2. **Get Joint Probabilities:** For each each class in  $y$ , compute the joint probability:  $\prod_{i=1}^d P(x_i | y)$
3. **Calculate Postiers:** Multiply our joint probabilities by the priors, (convert to logspace for ease of computation):

$$P(y | x) \propto P(y) \prod_{i=1}^d P(x_i | y)$$
$$\Rightarrow \log(Py | x) \propto \log P(y) + \log \sum_{i=1} P(x_i | y)$$

4. **Normalize Probabilities:** Convert the joint probabilities into valid probabilities by normalizing them to sum to 1.
5. **Assign Class:** Select the class  $y$  with the highest posterior probability as the prediction.

# Assumptions

We assume that features follow normal distributions. So, Gaussian Naive Bayes Classification works best when:

1. Features are not strongly correlated.
2. Data is exclusively continuous.
3. Outliers, if present, do not significantly affect the means and variances the data.
4. Datasets are not relatively large.

In any of these cases, more complex models may be more successful.

# Machine Learning Algorithm

---

**Algorithm 1** Training the Gaussian Naive Bayes Model

---

**Input:**  $X_{\text{train}}$  (2D:  $n_{\text{examples}} \times n_{\text{attributes}}$ ),  $y_{\text{train}}$  (1D:  $n_{\text{examples}}$ ),  $\text{classes} = \{c_1, \dots, c_k\}$  (1D:  $n_{\text{classes}}$ )

Set  $n_{\text{attributes}}$  = number of columns in  $X_{\text{train}}$

Initialize  $\mu_{\text{class}}$  as a zero matrix of size  $(n_{\text{classes}} \times n_{\text{attributes}})$

Initialize  $\sigma_{\text{class}}^2$  as a zero matrix of size  $(n_{\text{classes}} \times n_{\text{attributes}})$

**for** each  $c \in \text{classes}$  **do**

$\text{priors}[c]$  = fraction of instances of class in  $y_{\text{train}}$

    Set  $X_c$  = all  $x_i \in X_{\text{train}}$  belonging to class  $c$

**for** each attribute  $j$  in  $x_i$  **do**:

$X_{c,j}$  = the  $j$ th component of all  $x_i \in X_c$

        Set  $\mu_{\text{class}}[c, j]$  = mean of  $X_{c,j}$

        Set  $\sigma_{\text{class}}^2[c, j]$  = variance of  $X_{c,j}$

**end for**

**end for**

**Return:**  $\text{priors}$ ,  $\mu_{\text{class}}$ ,  $\sigma_{\text{class}}^2$

---

# Machine Learning Algorithm

---

**Algorithm 2** Predict Class for Given Examples

---

**Input:**  $X_{\text{test}}$  (2D:  $n_{\text{examples}} \times n_{\text{attributes}}$ ),  $\mu_{\text{class}}$  (1D:  $n_{\text{classes}}$ ),  $\sigma_{\text{class}}^2$  (1D:  $n_{\text{classes}}$ ), priors (1D:  $n_{\text{classes}}$ )

**for** each example  $x_i$  in  $X_{\text{test}}$  **do**

Set  $\epsilon = 1 \times 10^{-10}$

**for** each class  $c \in \text{classes}$  **do**

log\_prior = log(priors[c])

$\sigma^2 = \max(\sigma_{\text{class}}^2[c], \epsilon)$  (to avoid division by 0)

$\mu = \mu_{\text{class}}[c]$

$$\text{log\_likelihood} = -\frac{1}{2} \sum_{j=1}^{n_{\text{attributes}}} \left( \log(2\pi\sigma_j^2) + \frac{(x_{ij} - \mu_j)^2}{\sigma_j^2} \right)$$

posteriors[c] = log\_prior + log\_likelihood

**end for**

predictions[i] = arg max(posteriors) (class with the highest posterior probability)

**end for**

**Return:** predictions

---

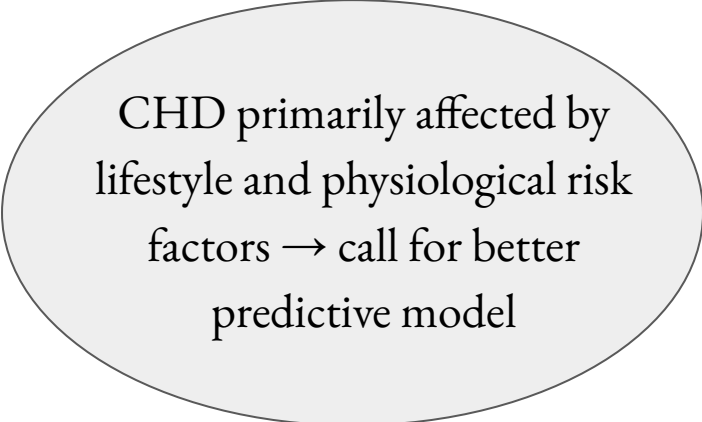


# Machine-Learning-Based Prediction Models of Coronary Heart Disease Using Naïve Bayes and Random Forest Algorithms

August 2021

DOI:[10.1109/ICSECS52883.2021.00049](https://doi.org/10.1109/ICSECS52883.2021.00049)

Conference: 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)



CHD primarily affected by lifestyle and physiological risk factors → call for better predictive model

## UCI Repository - Cleveland Database

- 303 records
- 76 factors overall - 13 risk factors analyzed
- Categorical and continuous variables

**Accuracy** = 85%

# Implementation and Results

## Data pre-processing:

- Convert non-numerical categorical features into numerical representations
- Remove missing/null values

## Data training:

Still 303 records

80/20 split

1. Standard split
2. Train\_test\_split method

## Our model:

1. 73.77%
2. 85.25%

## Sklearn:

1. 73.77%
2. 85.25%

# Implementation and Results

## Data pre-processing:

- Convert non-numerical categorical features into numerical representations
- Remove missing/null values

## Data training:

Still 303 records

80/20 split

1. Standard split
2. Train\_test\_split method

## Our model:

1. 73.77%
2. 85.25%

## Sklearn:

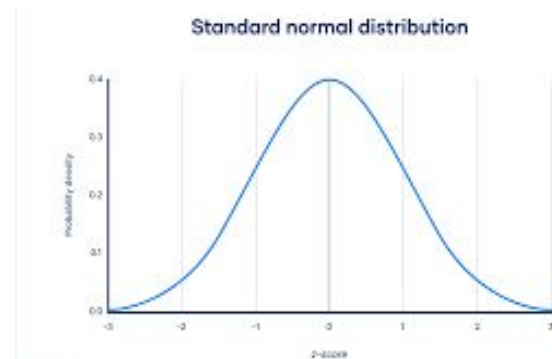
1. 73.77%
2. 85.25%



**Paper: 85%**

# Why We Liked Our Algorithm

- Gaussian Naive Bayes (GNB) offers several advantages that were great for our datasets (and ones alike)
  - Our dataset contains continuous attributes, and GNB is well-suited for handling such data
  - It assumes that the features follow a Gaussian distribution
  - GNB can perform well even with relatively small sample sizes. Since the Cleveland dataset consists of 303 records, GNB is advantageous
  - GNB is computationally efficient



# Comparing To Other Algorithms

- When looking at other potential algorithms, the two that were considered and discussed in the paper were Bernoulli Naïve Bayes and Random Forest algorithms
- Bernoulli Naive Bayes is simple to implement and gives good accuracy with small datasets. It is particularly efficient for binary datasets and text classification tasks - however, it is less suitable for our dataset due to the continuous nature of the data and the assumption of binary features, which does not align with the it's characteristics
- Random Forest can handle complex interactions and high-dimensional data - however, Random Forest can suffer from overfitting, especially when working with small datasets.

# Challenging As We Implemented It

- Actually implementing the algorithm posed little issues
- Testing the GNB algorithm to match the paper posed issues
- The paper is explicit about how it is training the model (uses train-test data split of 80:20, has an alpha of  $1e-9$ ). However, it was unclear how the data was split (random or not, and how random it was) - required experimentation to match the accuracy of the paper
- Tested against SKlearn with and without randomization to ensure our GNB matches its accuracy

# Sources

Baladram, S. (2024) *Gaussian Naive Bayes, Explained: A Visual Guide with Code Examples for Beginners, Medium*. Available at: <https://towardsdatascience.com/gaussian-naive-bayes-explained-a-visual-guide-with-code-examples-for-beginners-04949cef383c> (Accessed: 10 December 2024).

Bemando, Miranda, and Aryuni (2021) 'Machine-Learning-Based Prediction Models of Coronary Heart disease Using Naïve Bayes and Random Forest Algorithms,' *IEEE* Available at: <https://ieeexplore.ieee.org/document/9537060> (Accessed 12 December 2024)

*Gaussian Naive Bayes* (2023) *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/gaussian-naive-bayes/> (Accessed: 10 December 2024).

*GaussianNB* (no date) *scikit-learn*. Available at: [https://scikit-learn/stable/modules/generated/sklearn.naive\\_bayes.GaussianNB.html](https://scikit-learn/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html) (Accessed: 10 December 2024).

Kashishdafe (2024) 'Gaussian Naive Bayes: Understanding the Basics and Applications', *Medium*, 23 March. Available at: <https://medium.com/@kashishdafe0410/gaussian-naive-bayes-understanding-the-basics-and-applications-52098087b963> (Accessed: 10 December 2024).

'Naive Bayes classifier' (2024) *Wikipedia*. Available at: [https://en.wikipedia.org/w/index.php?title=Naive\\_Bayes\\_classifier&oldid=1260034546](https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=1260034546) (Accessed: 10 December 2024).