Project n^{o}1: Comparing Performance of Learning Algorithms for Heart Disease Detection Advanced Machine Learning (MDS)

Volpi Davide D'Agostino Raffaele Clerici Federico

Date: 17 November 2025

Abstract

We address the problem of heart disease detection using supervised learning methods on the UCI Heart Disease dataset. Our study compares the performance of four classification algorithms: Logistic Regression, Support Vector Machine, Custom Naive Bayes, and Decision Tree; using a rigorous evaluation protocol with repeated train-test splits and cross-validated hyperparameter optimization. We introduce a Custom Naive Bayes classifier tailored for mixed feature types and distributions. Experimental results show that linear discriminative models (Logistic Regression and SVM) achieve the highest and most stable AUC-ROC scores, while Decision Trees underperform. The Custom Naive Bayes model provides competitive results but does not surpass linear methods. Statistical analysis confirms significant differences between models. Our findings highlight the effectiveness of linear classifiers for this medical diagnosis task.

Note: all the notation and abbreviations that will be used can be found in Table 8.

1 Introduction

This work provides a robust framework for evaluating the performance of different algorithms for heart disease classification, a critical medical diagnosis task in which cardiovascular diseases remain the leading cause of mortality worldwide. We apply supervised learning methods to the widely known UCI *Heart Disease* dataset [Janosi(1989)].

We evaluate four classification approaches: LR and a CNB implementation (Part I), alongside Decision Trees and SVMs (Part II). We establish a rigorous evaluation protocol using cross-validation with systematic hyperparameter optimization, providing performance estimates alongside computational efficiency analysis.

Our main contributions include: (i) a CNB implementation for handling mixed probability distributions characterizing our data, (ii) a comprehensive cross-validation framework comparing four methods from different course modules, and (iii) joint analysis of generalization performance and the relation to the theoretical characteristics of the models.

The rest of this report is organized as follows: Section 2 states the problem. Section 3 reviews related work on the this dataset and analogous problems and establishes the problem context. Section 4 describes the data characteristics, preprocessing pipeline, and exploratory analysis. Section 5 details our experimental protocol and model configurations. Section 6 presents comparative results, examining generalization performance and computational trade-offs across methods.

2 Problem statement

We consider the supervised binary classification problem. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where each instance $\mathbf{x}_i \in \mathbb{R}^p$ represents a patient's clinical measurements across p features and $y_i \in \{0, 1\}$ denotes the corresponding class label (0: absence of heart disease, 1: presence of heart disease), our goal is to learn a classification function $f(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^p \to \{0, 1\}$, parameterized by $\boldsymbol{\theta}$, that accurately predicts the disease status of unseen patients.

Formally, we seek to minimize the expected misclassification risk:

$$R(f) = \mathbb{E}_{(\mathbf{x},y)\sim P}\left[\mathbb{1}_{[f(\mathbf{x};\boldsymbol{\theta})\neq y]}\right],$$

where P is the unknown underlying distribution over feature-label pairs, and $\mathbb{1}_{[\cdot]}$ is the indicator function. Since P is unknown, we approximate this by minimizing the empirical risk on $\mathcal{D}$:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[f(\mathbf{x}_i;\boldsymbol{\theta})\neq y_i]}$$

To address this learning problem effectively, we evaluate classifiers from multiple hypothesis classes. The dataset presents two key challenges: (i) mixed feature types (continuous and categorical variables), and (ii) potential mixed feature distributions within each class, which may violate algorithms' parametric assumptions.

3 Related Work

The UCI Heart Disease dataset, introduced by [Detrano(1989)], is a widely used benchmark for evaluating classification algorithms in medical diagnosis, which makes it difficult to find studies that focus exclusively on this dataset.

The original work established LR as a baseline, achieving an accuracy of 78.7%, and subsequent research has explored a range of supervised learning approaches on this benchmark. [Boer(2023)] compared LR, GNB, Random Forest, and SVM, reporting the best performance with GNB (79.79%) while also highlighting the method's limitations when feature distributions deviate from normality. [Syafi'ah(2025)] evaluated SVMs with various kernels, obtaining 83–86% accuracy depending on the kernel choice. Broader comparative studies [Sadikoglu(2022), Malik(2021)] consistently rank LR, SVM, and ensemble methods among the most effective models for binary medical classification tasks, while Cherian and Bindu [Cherian and Bindu(2017)] demonstrate that Naive Bayes with Laplace smoothing can also achieve competitive performance for heart disease prediction.

Our work differs from previous approaches by implementing a CNB variant that handles mixed feature distributions through separate density estimation for numerical and categorical features, addressing standard GNB limitations. Additionally, we provide systematic hyperparameter optimization via cross-validation to ensure fair comparison across methods.

4 Data and Preprocessing

4.1 Data description

The UCI Heart Disease Dataset in use has been introduced by [Detrano(1989)] and is stored in the UCI repository [Janosi(1989)]. The goal field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. We follow the standard practice in experiments with the Cleveland database, focusing on simply attempting to distinguish presence (values 1, 2, 3, 4) from absence (value 0). It contains a mix of continuous features (*age*, *trestbps*, *chol*, *thalach*, and *oldpeak*) and categorical features (*sex*, *cp*, *fbs*, *restecg*, *exang*, *slope*, *ca*, and *thal*). These independent features refer to vital parameters measured in a medical context, including demographics, blood pressure, cholesterol levels, electrocardiographic results, and cardiac stress test outcomes.

The original data present diverse challenges, such as the presence of missing values in the *ca* (number of major vessels) and *thal* (thalassemia type) features, and potential outliers in continuous measurements. The binary target classes are relatively balanced, with approximately 54.5% presence and 45.5% absence.

4.2 Exploratory data analysis

We begin by assessing missingness and find 6 observations with missing values concentrated in *ca* and *thal*. We proceed by removing the corresponding rows due to the low proportion (2%).

Next, we examine class-conditional distributions of continuous variables (*age*, *trestbps*, *chol*, *thalach*, *oldpeak*) using histograms with KDE overlays and QQ-plots (Figure 1). The per-class

Table 1: UCI Heart Disease Dataset characteristics

Property	**Variable**	**Value**
Number of observations	n	303
Number of features	d	13
Feature types	continuous, categorical	/
Number of classes	C (for classification)	2
Class distribution	balanced/imbalanced	54.5%/45.5%
Rows with missing values	percentage	2%

density and QQ diagnostics indicate approximate normality for most variables, with visible deviations in the tails and a few extreme observations; applying the IQR rule ($1.5 \times$ IQR), we identify 20 outliers that will be addressed during preprocessing. Among the continuous variables, *oldpeak* exhibits marked right-skewness with exponential-like decay, suggesting potential benefit from robust scaling or monotonic transformations. Class proportions appear reasonably balanced in the per-class histograms (except for *oldpeak*), reducing the need for aggressive rebalancing strategies.

Finally, we explore associations via a correlation matrix for continuous features (Figure 1). The observed patterns indicate medium to low correlations among the independent variables, with no evidence of problematic collinearity (maximum pairwise correlation < 0.5).

Figure 1: Distribution analysis (left) and Correlation matrix (right)

4.3 Preprocessing steps

We apply model-specific preprocessing to address data characteristics identified in exploratory analysis while respecting each algorithm's assumptions.

We identify 6 observations (2%) with missing values in *ca* and *thal* features. Given the small proportion, we apply complete case deletion, removing these observations to avoid imputation bias.

We retain the 20 observations flagged as outliers by the IQR rule because, in cardiovascular datasets, extreme values typically correspond to genuinely severe cases rather than noise, so removing them would discard clinically critical patterns. Moreover, the models considered in this work are reasonably robust to moderate outliers: decision trees base their splits on order statistics rather than distances, soft-margin SVMs with regularization constrain the influence of individual points, and regularized logistic regression and the CNB classifier are trained on standardized features, which collectively limits the impact of extreme values on the learned decision boundaries. Nonetheless, because LR and CNB are not inherently robust to outliers, we retain these extreme cases to assess whether they degrade the predictive performance.

For methods assuming Gaussian distributions (CNB), we apply Box-Cox transformation [Box and Cox(1964)] to continuous features to reduce skewness and reduce the influence of outliers

on normality. The optimal λ parameter is estimated via maximum likelihood for each feature independently.

Following transformation, we apply standardization (zero mean, unit variance) using sklearn's `StandardScaler` to continuous features for LR, SVM, and CNB. This facilitates the gradient descent convergence (LR) and ensures balanced contributions in distance calculations (SVM), and improving numerical stability in covariance estimation (CNB).

We apply OHE to most of the categorical features (*cp*, *restecg*, *slope*, *ca*, *thal*) for LR and SVM, which require numerical inputs. We drop the first category of each variable to avoid multicollinearity. Decision Trees receive unencoded categorical features, as they handle them natively through split criteria. The binary variables (*sex*, *fbs* and *exang*) remain unchanged.

The binary target exhibits balanced distribution (54.5% presence, 45.5% absence), eliminating the need for resampling or class weighting strategies.

5 Methodology

5.1 Experimental protocol

Experiments employ repeated random subsampling validation with 100 iterations. In each iteration, the dataset is stratified and split into training (75%) and test (25%) sets to preserve class balance. The training data are used for hyperparameter tuning via 10-fold cross-validation and final model fitting, while the test set is reserved exclusively for final evaluation. Averaging results across iterations yields stable and robust performance estimates.

The primary evaluation metric is the AUC-ROC, chosen for its robustness to class imbalance and its threshold-independent assessment of discriminative performance [Hanley(1982)].

Hyperparameters are tuned first by doing a telescopic search to select the appropriate range, then via grid search on the selected range with 10-fold cross-validation on the training set. The optimal configuration is selected based on the highest mean AUC-ROC, and this process is repeated for each of the 100 random splits.

All preprocessing and model selection steps are confined to the training data.

Results are reported as mean AUC-ROC $\pm$ standard deviation across 100 iterations. This resampling-based evaluation captures both central tendency and variability, providing statistically meaningful comparisons.

To assess whether observed differences in mean AUC-ROC between models are statistically significant, we applied paired t-tests to the AUC scores from 100 repeated random splits. P-values were adjusted using Bonferroni correction [Bonferroni(1936)] to control the family-wise error rate across all pairwise model comparisons, ensuring robust conclusions regarding relative model performance.

For each model and for each of the 100 runs, we then compute the ROC curve and select the optimal decision threshold using *Youden's J index* [Youden(1950)], defined as the threshold that maximizes $\mathrm{TPR}(t) + \mathrm{TNR}(t) - 1$. Using this threshold, we convert the predicted scores into binary labels and record all misclassifications. Finally, we aggregate results across the 100 runs and report the Top-10 most frequently misclassified instances for each model (Appendix A), allowing us to identify systematically difficult cases and make further analysis.

Experiments are implemented in Python; exact library versions are listed in `requirements.txt`. Computations were run on ASUS Zenbook 14 (Intel i7, 11th Gen) and two MacBook Air (M1/M2).

5.2 Method 1: Regularized Logistic Regression

5.2.1 Model formulation

The model minimizes the regularized negative log-likelihood (cross-entropy loss [Bishop(2006)]) of the LR:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \left\{ -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \sigma(x_i^\top \theta) + (1 - y_i) \log(1 - \sigma(x_i^\top \theta)) \right] + \lambda R(\theta) \right\} \tag{1}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function, $R(\theta)$ is a regularization term (typically ℓ_2-norm, $R(\theta) = \frac{1}{2}\|\theta\|_2^2$), and $\lambda > 0$ controls the regularization strength. Its decision boundary defined by

$$\{x : x^\top \theta = 0\} \tag{2}$$

5.2.2 Theoretical properties and justification

Under mild regularity conditions, the MLE $\hat{\theta}$ is consistent and asymptotically normal. The optimization problem is convex in θ, guaranteeing convergence to the global optimum.

The training complexity of LR is approximately $\mathcal{O}(nd^2)$ for n samples and d features. Prediction is $\mathcal{O}(d)$ per instance. With approximately 300 samples and 20 features (after encoding), training is computationally efficient, requiring only milliseconds per model fit.

We expect LR to perform competitively when the data exhibit predominantly linear or monotonic relationships between features and disease risk. Regularization helps control overfitting, especially when features are correlated (e.g., age and maximum heart rate). However, the model may underperform if strong non-linear interactions exist between clinical measurements, such as threshold effects or complex feature combinations.

For the full implementation details see Appendix C.1.

5.3 Method 2: Custom Naive Bayes (CNB)

5.3.1 Model formulation

The Custom Naive Bayes (CNB) classifier is designed to handle heterogeneous feature types by combining different probabilistic models tailored to the specific distributional characteristics of each variable group. Given an input feature vector $\mathbf{x}$ and a binary class label $y \in \{0, 1\}$, the classifier computes the posterior probability $P(y \mid \mathbf{x})$ using Bayes' theorem:

$$P(y \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid y)P(y)}{P(\mathbf{x})} \propto P(\mathbf{x} \mid y)P(y) \tag{3}$$

where $P(y)$ is the prior probability of class y, and $P(\mathbf{x} \mid y)$ is the class-conditional likelihood.

The feature vector $\mathbf{x}$ is partitioned into four distinct subsets:

$$\mathbf{x} = (\mathbf{x}_{\text{gauss}}, \mathbf{x}_{\text{bin}}, \mathbf{x}_{\text{cat}}, x_{\text{ng}}) \tag{4}$$

where $\mathbf{x}_{\text{gauss}} = (\text{age}, \text{trestbps}, \text{chol}, \text{thalach})^\top \in \mathbb{R}^4$ are continuous Gaussian features, $\mathbf{x}_{\text{bin}} = (\text{sex}, \text{fbs}, \text{exang})^\top \in \{0, 1\}^3$ are binary features, $\mathbf{x}_{\text{cat}} = (\text{cp}, \text{restecg}, \text{thal}, \text{slope}, \text{ca})$ are categorical features, and $x_{\text{ng}} = \text{oldpeak} \in \mathbb{R}$ is a continuous non-Gaussian feature.

Under the naive Bayes conditional independence assumption, the likelihood decomposes as:

$$P(\mathbf{x} \mid y) = P(\mathbf{x}_{\text{gauss}} \mid y) \cdot P(\mathbf{x}_{\text{bin}} \mid y) \cdot P(\mathbf{x}_{\text{cat}} \mid y) \cdot P(x_{\text{ng}} \mid y) \tag{5}$$

with further factorization for categorical and binary features:

$$P(\mathbf{x}_{\text{cat}} \mid y) = \prod_{x_j \in \mathbf{x}_{\text{cat}}} P(x_j \mid y) \qquad P(\mathbf{x}_{\text{bin}} \mid y) = \prod_{x_k \in \mathbf{x}_{\text{bin}}} p_{k,y}^{x_k}(1 - p_{k,y})^{1-x_k} \tag{6}$$

The class assignment rule is:

$$\hat{y} = \arg\max_{y \in \{0,1\}} \left[\log P(y) + \log P(\mathbf{x}_{\text{gauss}} \mid y) + \sum_{x_k \in \mathbf{x}_{\text{bin}}} \log P(x_k \mid y) + \sum_{x_j \in \mathbf{x}_{\text{cat}}} \log P(x_j \mid y) + \log P(x_{\text{ng}} \mid y) \right] \tag{7}$$

where logarithms are used for numerical stability.

5.3.2 Theoretical properties and justification

Each feature group is modeled with a distribution appropriate to its statistical properties, ensuring optimal representation of the underlying data-generating process.

Continuous Approximately Gaussian features: The features $\mathbf{x}_{\text{gauss}}$ are jointly modeled using a multivariate Gaussian distribution (after applying the necessary transformations to correct skewness):

$$P(\mathbf{x}_{\text{gauss}} \mid y) = \mathcal{N}(\mathbf{x}_{\text{gauss}} \mid \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_y|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_{\text{gauss}} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{x}_{\text{gauss}} - \boldsymbol{\mu}_y)\right) \tag{8}$$

where $d = 4$, and the class-specific parameters $\boldsymbol{\mu}_y \in \mathbb{R}^4$ and $\boldsymbol{\Sigma}_y \in \mathbb{R}^{4\times4}$ are estimated via maximum likelihood from the training data. This choice is justified by the Gaussianity observed in the empirical distributions of these features.

Binary features: Each binary feature follows an independent Bernoulli distribution with success probabilities estimated using Laplace smoothing:

$$p_{k,y} = \frac{\sum_{i:y_i=y} x_{k,i} + \alpha}{N_y + 2\alpha} \tag{9}$$

where N_y is the number of training samples with class y, and $\alpha = 1$ is a smoothing parameter that prevents zero probabilities for unseen feature combinations.

Categorical features: Discrete probability distributions are estimated empirically with additive smoothing:

$$P(x_j = c \mid y) = \frac{N_{j,c,y} + \alpha}{N_y + \alpha \cdot |C_j|} \tag{10}$$

where $N_{j,c,y}$ counts training samples with class y and feature j equal to category c, and $|C_j|$ is the cardinality of the categorical domain for feature j.

Continuous non-Gaussian feature: The `oldpeak` variable exhibits significant deviation from Gaussianity. Therefore, we employ nonparametric kernel density estimation (KDE):

$$P(x_{\text{ng}} \mid y) = \frac{1}{N_y h} \sum_{i:y_i=y} K\left(\frac{x_{\text{ng}} - x_{\text{ng},i}}{h}\right) \tag{11}$$

where $K(\cdot)$ is a Gaussian kernel function, h is the bandwidth parameter selected via Silverman method [Silverman(2018)], and $x_{\text{ng},i}$ are the training samples of `oldpeak` with class y. This approach provides a flexible, density estimate without restrictive parametric assumptions.

Decision boundary: The CNB classifier assigns classes by comparing the log-likelihoods in Equation (7), resulting in a generally *non-linear* decision boundary. The multivariate Gaussian term contributes a quadratic form, analogous to the boundary of QDA, while the Bernoulli and categorical components add linear terms in the log-odds. The KDE term introduces a locally adaptive, nonparametric contribution. Overall, the CNB boundary provides a flexible yet computationally simple classifier.

The CNB model relies on two key assumptions:

1. Conditional independence: Features are assumed independent given the class label (Equation 5). While this naive assumption is often violated in practice, it dramatically reduces model complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ parameters and often yields competitive classification performance despite model misspecification.

2. Distributional assumptions: Each feature group follows its specified distribution. We validated Gaussianity for $\mathbf{x}_{\text{gauss}}$ using Q-Q plots and visual inspection, and confirmed non-Gaussianity of `oldpeak` through the same diagnostic tools, justifying the KDE approach.

Our mixed Naive Bayes classifier combines multiple probability distributions to model different feature types: a multivariate Gaussian distribution for d_g continuous features (with full covariance matrix), d_b binomial distributions, d_c categorical distributions, and d_{kde} features modeled using Kernel Density Estimation (KDE) with Gaussian kernels.

The training phase complexity is:

$$\mathcal{O}\left((d_g^2 + d_b + d_c + d_{\text{kde}}) \cdot n + d_g^3 \cdot c\right)$$

where n is the total number of training samples and c is the number of classes. The $d_g^2 \cdot n$ term arises from estimating the full covariance matrix Σ for the multivariate Gaussian, while the $d_g^3 \cdot c$ term accounts for the matrix inversion required for each class. The binomial and categorical features require $\mathcal{O}((d_b + d_c) \cdot n)$ to compute frequency tables, and KDE requires $\mathcal{O}(d_{\text{kde}} \cdot n)$ to store all training points.

For each test instance, the prediction complexity is:

$$\mathcal{O}\left(c \cdot \left(d_g^2 + d_b + d_c + d_{\text{kde}} \cdot n_k\right)\right)$$

where n_k is the average number of training samples per class. The dominant term for large datasets is $d_{\text{kde}} \cdot n_k$, as KDE requires evaluating the kernel function over all stored training points for each class. The d_g^2 term results from the matrix-vector multiplication $\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})$ when evaluating the multivariate Gaussian density.

Under the assumption that the conditional independence holds and distributional specifications are a good approximation, the CNB classifier is a maximum likelihood estimator and achieves the Bayes optimal error rate asymptotically as $n \to \infty$.

For the full implementation details see Appendix C.4.

5.4 Method 3: Decision Tree (DT)

5.4.1 Model formulation

A decision tree recursively partitions the feature space $\mathcal{X}$ into disjoint regions and assigns a class label to each region based on the training samples within it. At each internal node, the tree selects a feature and threshold that optimally splits the data according to an impurity criterion. For classification, the Gini index is used:

$$\text{Gini}(S) = 1 - \sum_{k=1}^{K} p_k^2 \tag{12}$$

where p_k is the proportion of samples in set S belonging to class k, and K is the number of classes. The optimal split at node t is chosen to maximize the information gain:

$$\Delta I(t) = \text{Gini}(S_t) - \sum_{j \in \{L,R\}} \frac{|S_j|}{|S_t|} \text{Gini}(S_j) \tag{13}$$

where S_L and S_R are the left and right child nodes after splitting. The tree construction terminates when a maximum depth is reached, a minimum number of samples is required for splitting, or all samples in a node belong to the same class.

5.4.2 Theoretical properties and justification

DT create axis-aligned decision boundaries by recursively partitioning the feature space. The predicted class probability for a sample x is the proportion of training samples of each class in the leaf node containing x:

$$P(y = k|x) = \frac{1}{|S_\ell|} \sum_{i \in S_\ell} \mathbb{1}[y_i = k] \tag{14}$$

where S_ℓ is the set of training samples in the leaf node ℓ containing x.

DT make minimal distributional assumptions about the data, which makes them robust to various data characteristics. The model assumes that the true decision boundary can be approximated by axis-aligned hyperplanes, which may be inefficient for diagonal or curved boundaries but is reasonable for clinical thresholds (e.g. age $>$ 55, cholesterol $>$ 240).

DT are universal approximators that, given sufficient depth and training samples, they can approximate any decision boundary arbitrarily well. However, without regularization (limiting depth in our case), trees tend to overfit by memorizing the training data. The Gini impurity criterion ensures locally optimal splits at each node, though the global tree structure may be suboptimal due to the greedy nature of the construction algorithm.

Training complexity is $\mathcal{O}(nd \log n)$ for n samples and d features, as the algorithm must sort feature values to find optimal split points. Prediction is $\mathcal{O}(\text{depth})$ per instance, making inference extremely fast.

DT provide interpretable decision rules that can be visualized and validated by medical experts. However, they are prone to high variance: small changes in training data can lead to different tree structures, resulting in unstable predictions. On the heart disease dataset, we expect decision trees to effectively capture threshold-based patterns such as "$age > 60$ AND $cholesterol > 250 \rightarrow$ high risk" but may suffer from overfitting if depth is not carefully controlled. The maximum depth hyperparameter balances expressiveness (capturing complex patterns) with generalization (avoiding memorization of noise).

For the full implementation details go to Appendix C.2

5.5 Method 4: Support Vector Machine

5.5.1 Model formulation

A SVM for binary classification solves the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + CR(\xi) \quad \textit{subject to} \quad y_i(\mathbf{w}^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, \ldots, n \tag{15}$$

where $\mathbf{w}$ is the weight vector defining the decision hyperplane, b is the bias term, ξ_i are slack variables allowing misclassification, $C > 0$ is the regularization parameter controlling the trade-off between margin maximization and training error minimization, and $R(\xi)$ the penalization function. The decision function and decision boundary are defined by:

$$f(x) = \text{sign}(\mathbf{w}^\top x + b) \qquad \{x : \mathbf{w}^\top x + b = 0\} \tag{16}$$

5.5.2 Theoretical properties and justification

The SVM assumes that the data are approximately linearly separable in the feature space, or that slack variables can adequately handle non-separability, which is reasonable after standardization and one-hot encoding. The model also assumes that all features are on comparable scales, which is ensured through standardization in preprocessing. Independence of observations is assumed, which holds for individual patient records.

The SVM optimization problem is convex, guaranteeing convergence to the global optimum.

Training complexity for linear SVM is approximately $\mathcal{O}(n^2 d)$ for n samples and d features. Prediction is $\mathcal{O}(d)$ per instance.

We expect SVM to perform similarly to LR on this dataset, as both are linear discriminative models. However, SVM's computational cost is higher, and the model may struggle with highly imbalanced/overlapping class distributions, though the slack variables provide some robustness.

For the full implementation details see Appendix C.3

5.6 Comparison framework

Models are compared using mean AUC-ROC and its standard deviation across 100 repeated splits. This ensures a consistent, threshold-independent basis for comparison across heterogeneous models.

Significance is assessed by analyzing the distribution of AUC-ROC scores over 100 iterations. Non-overlapping confidence intervals ($\mu \pm \sigma$) indicate meaningful differences in model performance.

Beyond mean AUC-ROC, analyses include variability in optimal hyperparameters across iterations and standard deviation of AUC-ROC as a measure of sensitivity to data variation.

All models are trained and tested on identical stratified splits, using consistent tuning and evaluation protocols. Preprocessing is model-appropriate but always fitted on training data only. Probability outputs (`predict_proba`) are used for AUC-ROC computation to ensure comparability across architectures.

6 Discussion of results

6.1 Overall performance

Table 2 presents the comparative performance of all four methods across 100 iterations of stratified random splitting. Logistic Regression achieved the highest mean AUC-ROC (0.906 ± 0.003), followed closely by SVM (0.903 ± 0.003), then Custom Naive Bayes (0.897 ± 0.004), with Decision Trees performing substantially worse (0.818 ± 0.005). All methods exhibited relatively low standard deviations, indicating stable performance across different data splits.

To assess statistical significance, we conducted pairwise paired t-tests with Bonferroni correction (adjusted $\alpha = 0.0083$ for 6 comparisons). As shown in Table 3, all pairwise differences were statistically significant. The largest performance gaps were between linear methods (LR/SVM) and Decision Trees (mean differences > 0.08, both $p < 10^{-6}$), while differences among LR, SVM, and CNB were smaller but still significant (mean differences < 0.01, all $p < 0.008$).

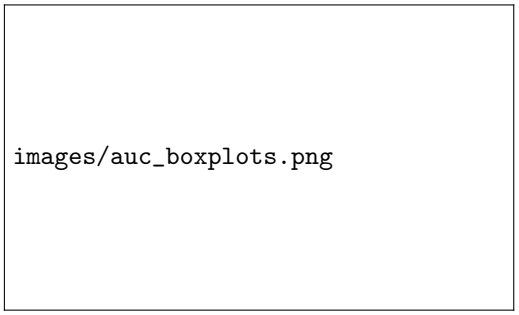
Logistic Regression emerged as the best performer, achieving both the highest AUC-ROC and the fastest training time (0.539 ± 0.006 ms). This is 7 times shorter than SVM and 11 times

shorter than CNB while maintaining superior predictive performance. Figure 2 visualizes the AUC distributions (left) and optimal hyperparameter selections (right) across all iterations, confirming the stability of these findings.

We identified the most misclassified observations for each model over the 100 experiments (detailed results in Appendix A).

Table 2: Performance comparison of methods on test set. Results shown as mean ± standard deviation over 100 rounds. Bold values indicates best performance. (Times obtained on the M1 MacBook Air)

Method	**AUC**	**Training time (ms)**
Logistic Regression	**0.906 ± 0.003**	**0.539± 0.006**
Custom Naive Bayes	0.897 ± 0.004	6.284 ± 0.671
Decision Tree	0.818 ± 0.005	0.583 ± 0.012
Support Vector Machine	0.903 ± 0.003	3.848 ± 0.094



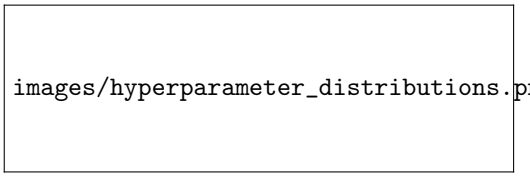


Figure 2: AUC distributions for each model (left) and Hyperparameters for each model (right)

Table 3: Pairwise statistical comparison of models using paired t-test with Bonferroni correction ($\alpha = 0.0083$).

Comparison	**Mean Diff.**	**Std. Err.**	***t*-stat**	***p*-value**	**Significant**
LR vs CNB	0.0093	0.0222	4.1878	0.000061	Yes
LR vs DTs	0.0884	0.0418	21.1523	$< 10^{-6}$	Yes
LR vs SVM	0.0038	0.0119	3.1834	0.001946	Yes
CNB vs DT	0.0791	0.0462	17.1052	$< 10^{-6}$	Yes
CNB vs SVM	−0.0055	0.0201	−2.7370	0.007351	Yes
DT vs SVM	−0.0846	0.0415	−20.3570	$< 10^{-6}$	Yes

6.2 Detailed analysis

As already discussed in Section 6.1, Logistic Regression (LR) and SVM, achieve the best results in terms of average AUC and low variance. This suggests that the classes are sufficiently well separated that a linear decision boundary can closely approximate the underlying class separation, which in turn confers an advantage to discriminative models that explicitly learn such separating hyperplanes.

The Custom Naive Bayes (CNB), despite its more complex structure and ability to model non-linear class boundaries as a generative model, performs slightly worse than LR and SVM. This is likely due to the additional complexity not being fully justified by the data, combined with the independence and distributional assumptions of our CNB that may not hold perfectly in this dataset. On the other hand, the Decision Tree (DT), which in principle can approximate any arbitrary decision function, is the worst-performing model. This is consistent with the known tendency of DTs to overfit small datasets, as they can create highly specific splits that do not generalize well. Overall, these results indicate that linear, discriminative models fit this dataset effectively, while

more flexible models like CNB and DT do not offer additional predictive advantages and may even underperform due to overfitting.

A pairwise paired t-test with Bonferroni correction confirms that the linear models (LR and SVM) significantly outperform both CNB and DT, with DT showing the largest performance gaps.

Figure 2 shows the distributions of selected hyperparameters across 100 iterations. LR favors higher regularization values ($C \approx 1.5$–2.0), indicating minimal overfitting risk. DT predominantly selects maximum depth $= 3$, reflecting the need for strong regularization on small folds. SVM's C parameter distribution seems to suggest lower values of C. The observed difference in regularization preferences can be explained by the models' underlying mechanisms. SVM's hinge loss and margin maximization make it more sensitive to individual support vectors and prone to overfitting on small training folds, thus requiring stronger regularization (lower C). In contrast, LR's smooth probabilistic loss naturally produces softer decision boundaries that are less susceptible to overfitting, allowing for weaker regularization (higher C). This suggests that the dataset is sufficiently clean and linearly separable for LR to perform well with minimal penalty, while SVM benefits from additional constraints to prevent fitting noise in the support vectors.

To investigate model failures beyond aggregate metrics (See Appendix A for complete results), we analyze the top 10 most frequently misclassified samples across 100 experimental runs. Three samples (145, 250, 301) are consistently misclassified by all four models, representing inherently ambiguous boundary cases. The class distribution within errors reveals systematic biases: DT exhibits the strongest false negative tendency (80% diseased patients misclassified in the top 10 misclassified items), followed by CNB (70%), SVM (60%), and balanced LR (50%). This pattern indicates that DT's greedy splitting and CNB's conditional independence assumption both fail to capture complex risk factor interactions in diseased patients. Feature analysis shows asymptomatic chest pain (cp=4) appears in 40–50% of all models' hard cases, while DT uniquely struggles with exercise-induced angina (exang=1 in 40% of errors vs. 20% for others), suggesting depth-3 trees inadequately partition continuous feature interactions. Notably, DT's errors have the lowest mean maximum heart rate (thal=141.8) and highest ST depression (oldpeak=0.82), indicating difficulty with severe continuous markers. Conversely, SVM and CNB show lower oldpeak values in errors (0.52 and 0.55), struggling with borderline cases lacking extreme indicators. These complementary failure modes confirm that while linear discriminative models handle this dataset's structure effectively, added complexity from probabilistic modeling or hierarchical partitioning introduces model-specific biases without systematic performance gains.

7 Conclusions

In this project we address heart disease prediction using the UCI dataset, evaluating Logistic Regression, Support Vector Machine, a Custom Naive Bayes, and a Decision Tree via repeated cross-validated AUC. LR and SVM achieve the highest and most stable performance, indicating that the underlying decision boundary is approximately linear. The CNB, which models continuous features with a multivariate Gaussian, assumes independence for categorical features, and uses a KDE component for a non-Gaussian variable, also performed well. However, its additional flexibility does not yield a clear performance advantage over the linear discriminative models. DTs exhibited the lowest and most variable AUC, reflecting their tendency to overfit small datasets despite the use of regularization techniques. Overall, the results show that linear discriminative models are sufficient for this dataset, while generative modeling through CNB can achieve competitive results without being strictly necessary. Theoretical considerations align with these observations: generative models perform well when distributional assumptions are reasonable, and high-variance models like DTs are unstable with limited data. Our study focuses on performance and robustness, running 100 different train-test splits to ensure statistical reliability. This approach limited our ability to explore additional aspects such as (i) sensitivity to hyperparameters, (ii) training dynamics including convergence and learning curves, and (iii) model interpretability, e.g., feature importance or visualization of decision boundaries. Future work could address these analyses to gain deeper insights into model behavior and practical interpretability.

8 Declarations on GenAI

The authors would like to acknowledge the use of ChatGPT [OpenAI(2025)] for assistance with spell checking, debugging, and generating well-formatted plots. The tool was used solely to improve clarity and presentation; all analysis and results were performed independently by the authors.

References

[Bishop(2006)] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 1st edition, 2006.

[Boer(2023)] Yoshiven et al. Boer. Classification of heart disease: Comparative analysis using knn, random forest, gaussian naive bayes, xgboost, svm, decision tree, and logistic regression. In *2023 5th International Conference on Cybernetics and Intelligent System (ICORIS)*, pages 1–5. IEEE, 2023.

[Bonferroni(1936)] Carlo E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8: 3–62, 1936.

[Box and Cox(1964)] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243, 1964.

[Cherian and Bindu(2017)] Vincy Cherian and MS Bindu. Heart disease prediction using naive bayes algorithm and laplace smoothing technique. *Int. J. Comput. Sci. Trends Technol*, 5(2): 68–73, 2017.

[Detrano(1989)] Robert et al. Detrano. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.

[Fan(2008)] Rong-En et al. Fan. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. URL https://www.jmlr.org/papers/volume9/fan08a/fan08a.pdf.

[Hanley(1982)] James A. et al. Hanley. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[Janosi(1989)] Andras et al. Janosi. Heart disease. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C52P4X.

[Malik(2021)] Munjal Malik. Reviewing classification methods on health care. In *Intelligent Healthcare: Applications of AI in eHealth*, pages 127–142. Springer, 2021.

[OpenAI(2025)] OpenAI. Chatgpt (november 17 version). [Large language model] https://chat.openai.com/, 2025.

[Pedregosa(2011)] Pedregosa. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[Python Software Foundation(2022)] Python Software Foundation. Python 3.11.0 release. https://www.python.org/downloads/release/python-3110/, 2022.

[Sadikoglu(2022)] Fahreddin et al. Sadikoglu. Performance analysis of machine learning algorithms for medical datasets. In *International Conference on Theory and Applications of Fuzzy Systems and Soft Computing*, pages 514–521. Springer, 2022.

[Silverman(2018)] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[Syafi'ah(2025)] Nurus et al. Syafi'ah. Cross-dataset evaluation of support vector machines: A reproducible, calibration-aware baseline for tabular classification. *Jurnal Riset Mahasiswa Matematika*, 4(6):323–337, 2025.

[Youden(1950)] William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

A Additional results

Table 4: Top 10 most frequently misclassified samples for LR across 100 experimental runs.

ID	Count	%	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
301	30	30.0	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	1
218	30	30.0	64	0	4	130	303	0	0	122	0	2.0	2	2.0	3.0	0
252	29	29.0	64	1	4	128	263	0	0	105	1	0.2	2	1.0	7.0	0
275	28	28.0	64	1	1	170	227	0	2	155	0	0.6	2	0.0	7.0	0
33	27	27.0	59	1	4	135	234	0	0	161	0	0.5	2	0.0	7.0	0
145	27	27.0	47	1	3	108	243	0	0	152	0	0.0	1	0.0	3.0	1
22	26	26.0	58	1	2	120	284	0	2	160	0	1.8	2	0.0	3.0	1
261	26	26.0	58	0	2	136	319	1	2	152	0	0.0	1	2.0	3.0	1
16	26	26.0	48	1	2	110	229	0	0	168	0	1.0	3	0.0	7.0	1
250	25	25.0	57	1	4	110	201	0	0	126	1	1.5	2	0.0	6.0	0

Table 5: Top 10 most frequently misclassified samples for CNB across 100 experimental runs.

ID	Count	%	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
188	34	34.0	54	1	2	192	283	0	2	195	0	0.0	1	1.0	7.0	1
278	30	30.0	57	1	2	154	232	0	2	164	0	0.0	1	1.0	3.0	1
301	30	30.0	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	1
218	30	30.0	64	0	4	130	303	0	0	122	0	2.0	2	2.0	3.0	0
252	29	29.0	64	1	4	128	263	0	0	105	1	0.2	2	1.0	7.0	0
268	29	29.0	40	1	4	152	223	0	0	181	0	0.0	1	0.0	7.0	1
250	28	28.0	57	1	4	110	201	0	0	126	1	1.5	2	0.0	6.0	0
199	28	28.0	59	1	1	160	273	0	2	125	0	0.0	1	0.0	3.0	1
145	27	27.0	47	1	3	108	243	0	0	152	0	0.0	1	0.0	3.0	1
22	27	27.0	58	1	2	120	284	0	2	160	0	1.8	2	0.0	3.0	1

Table 6: Top 10 most frequently misclassified samples for DT across 100 experimental runs.

ID	Count	%	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
252	28	28.0	64	1	4	128	263	0	0	105	1	0.2	2	1.0	7.0	0
245	27	27.0	67	1	4	120	237	0	0	71	0	1.0	2	0.0	3.0	1
172	27	27.0	59	0	4	174	249	0	0	143	1	0.0	2	0.0	3.0	1
145	27	27.0	47	1	3	108	243	0	0	152	0	0.0	1	0.0	3.0	1
301	27	27.0	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	1
22	27	27.0	58	1	2	120	284	0	2	160	0	1.8	2	0.0	3.0	1
278	26	26.0	57	1	2	154	232	0	2	164	0	0.0	1	1.0	3.0	1
69	26	26.0	46	1	3	150	231	0	0	147	0	3.6	2	0.0	3.0	1
294	26	26.0	63	0	4	124	197	0	0	136	1	0.0	2	0.0	3.0	1
250	25	25.0	57	1	4	110	201	0	0	126	1	1.5	2	0.0	6.0	0

Table 7: Top 10 most frequently misclassified samples for SVM across 100 experimental runs.

ID	Count	%	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
301	30	30.0	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	1
252	29	29.0	64	1	4	128	263	0	0	105	1	0.2	2	1.0	7.0	0
33	28	28.0	59	1	4	135	234	0	0	161	0	0.5	2	0.0	7.0	0
199	28	28.0	59	1	1	160	273	0	2	125	0	0.0	1	0.0	3.0	1
218	28	28.0	64	0	4	130	303	0	0	122	0	2.0	2	2.0	3.0	0
145	27	27.0	47	1	3	108	243	0	0	152	0	0.0	1	0.0	3.0	1
250	26	26.0	57	1	4	110	201	0	0	126	1	1.5	2	0.0	6.0	0
261	26	26.0	58	0	2	136	319	1	2	152	0	0.0	1	2.0	3.0	1
278	25	25.0	57	1	2	154	232	0	2	164	0	0.0	1	1.0	3.0	1
232	24	24.0	49	1	3	118	149	0	2	126	0	0.8	1	3.0	3.0	1

B Notation

Table 8: Model abbreviations used throughout the text

Model	Abbreviation
Logistic Regression	LR
Support Vector Machine	SVM
Naive Bayes	NB
Custom Naive Bayes	CNB
Gaussian Naive Bayes	GNB
Decision Tree	DT
One-Hot Encoding	OHE
Area Under the Receiver Operating Characteristic Curve	AUC-ROC
Maximum Likelihood Estimator	MLE

C Implementation Details

C.1 Implementation details of LR

The LR is implemented using `python (v 3.11)` [Python Software Foundation(2022)] with the `scikit-learn (v 1.6.1)` [Pedregosa(2011)] library through the `LogisticRegression()` class. The primary hyperparameter tuned and selected are:

- `penalty='l2'`: Ridge regularization.
- `C`: inverse of the regularization strength ($C = 1/\lambda$).
- `solver`: `'liblinear'`, suitable for small datasets and L2-regularized binary classification [Fan(2008)].

For the additional fixed hyperparameters see Table 9.

C.2 Implementation Details of DT

The DT is implemented using `python (v 3.11)` [Python Software Foundation(2022)] with the `scikit-learn (v 1.6.1)` [Pedregosa(2011)] library through the `DecisionTreeClassifier()` class. The primary hyperparameter tuned is:

- `max_depth`: maximum depth of the tree.

For the additional fixed hyperparameters see Table 9.

C.3 Implementation details of SVM

The SVM is implemented using `python (v 3.11)` [Python Software Foundation(2022)] with the `scikit-learn (v 1.6.1)` [Pedregosa(2011)] library through the `SVC()` class. The primary hyperparameter tuned and selected are:

- `penalty='l2'`: Ridge regularization.
- `C`: regularization parameter controlling the trade-off between margin maximization and classification error minimization.
- `kernel`: `linear`, providing interpretable feature weights and computational efficiency.
- `Probability = True`, to enable probability estimates.

For the additional fixed hyperparameters see Table 9.

C.4 Implementation Details of CNB

Algorithm 1 Custom Naive Bayes

Require: Training data $X \in \mathbb{R}^{n \times d}$, labels $y \in \{0,1\}^n$
Require: Feature partitions: categorical $\mathcal{F}_{cat}$, binary $\mathcal{F}_{bin}$, oldpeak f_{old}, numerical $\mathcal{F}_{num}$
Ensure: Trained model parameters: π_k, θ_{cat}, θ_{bin}, KDE_k, μ_k, Σ_k

// **Training Phase**
Compute class priors: $\pi_k \leftarrow \frac{|\{i:y_i=k\}|}{n}$ for $k \in \{0,1\}$
// **Fit categorical features (with Laplace smoothing)**
for each feature $f \in \mathcal{F}_{cat}$ **do**
  for each class $k \in \{0,1\}$ **do**
    $X_k \leftarrow X[y=k, f]$
    $\theta_{cat}^{f,k}(v) \leftarrow \frac{\text{count}(X_k=v)+1}{|X_k|+|\text{unique}(f)|}$ for all values v
  end for
end for
// **Fit binary features (Bernoulli distribution)**
for each feature $f \in \mathcal{F}_{bin}$ **do**
  for each class $k \in \{0,1\}$ **do**
    $\theta_{bin}^{f,k} \leftarrow \frac{\sum_{i:y_i=k} X_{i,f}+1}{|\{i:y_i=k\}|+2}$ {$P(f=1|k)$ with Laplace}
  end for
end for
// **Fit oldpeak feature (Kernel Density Estimation)**
for each class $k \in \{0,1\}$ **do**
  $X_{old}^k \leftarrow X[y=k, f_{old}]$
  Compute bandwidth h_k using Silverman's rule: $h_k = 0.9 \cdot \min\{\sigma_{old}^k, \text{IQR}_{old}^k/1.34\} \cdot n_k^{-1/5}$
  Fit $KDE_k \leftarrow \text{GaussianKDE}(X_{old}^k, h_k)$
end for
// **Fit numerical features (Multivariate Gaussian)**
for each class $k \in \{0,1\}$ **do**
  $X_{num}^k \leftarrow X[y=k, \mathcal{F}_{num}]$ {Include age, trestbps, chol, thalach}
  Compute mean: $\mu_k \leftarrow \frac{1}{n_k} \sum_{i:y_i=k} X_{i,\mathcal{F}_{num}}$
  Compute covariance: $\Sigma_k \leftarrow \frac{1}{n_k} \sum_{i:y_i=k} (X_{i,\mathcal{F}_{num}} - \mu_k)(X_{i,\mathcal{F}_{num}} - \mu_k)^\top + \epsilon I$
  Compute Σ_k^{-1} and $|\Sigma_k|$ {For efficient prediction}
end for

// **Prediction Phase (for test sample x)**
for each class $k \in \{0,1\}$ **do**
  $\log P(y=k|x) \leftarrow \log \pi_k$ {Start with log prior}
  // **Add log-likelihoods from each feature type**
  $\log P(y=k|x) \leftarrow \log P(y=k|x) + \sum_{f \in \mathcal{F}_{cat}} \log \theta_{cat}^{f,k}(x_f)$
  $\log P(y=k|x) \leftarrow \log P(y=k|x) + \sum_{f \in \mathcal{F}_{bin}} [x_f \log \theta_{bin}^{f,k} + (1-x_f) \log(1-\theta_{bin}^{f,k})]$
  $\log P(y=k|x) \leftarrow \log P(y=k|x) + \log KDE_k(x_{old})$
  $\log P(y=k|x) \leftarrow \log P(y=k|x) - \frac{1}{2} \left[\log((2\pi)^d |\Sigma_k|) + (x_{num} - \mu_k)^\top \Sigma_k^{-1} (x_{num} - \mu_k)\right]$
end for
// **Normalize using log-sum-exp trick**
$m \leftarrow \max_k \log P(y=k|x)$
$P(y=k|x) \leftarrow \frac{\exp(\log P(y=k|x) - m)}{\sum_{k'} \exp(\log P(y=k'|x) - m)}$ for $k \in \{0,1\}$
return $P(y=1|x)$ {Probability of positive class}

C.5 Models Hyperparameters

Table 9: Hyperparameters for all evaluated models

Model	**Hyperparameter**	**Value / Search space**
Logistic Regression	C	$[10^{-9}, 2]$ (100 linearly-spaced values)
	solver	`liblinear`
	max_iter	1000
	random_state	42
Decision Tree	max_depth	$\{1, 2, 3, 4, 5, 6\}$
	criterion	Gini (by default)
	random_state	42
SVM	C	$[10^{-7}, 1]$ (100 logarithmically-spaced values)
	probability	True
	kernel	`linear`
	random_state	42