IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

Heart Disease Risk Prediction and Classification Using Machine Learning

Ms. Pallavi Devidas Ahinave[1] and Mrs. Shubhangi Pratik Bombale[2]
[1]Student Jaihind Institute of Management and Research ,Junnar, India
[2]Assistatmt Professor, Jaihind Institute of Management and Research , Junnar, India

Abstract: *Heart disease remains one of the leading causes of death worldwide, and early recognition of elevated risk can prompt people to seek timely clinical evaluation. This paper presents the design of a web-based decision-support demonstrator that estimates the likelihood of heart disease from thirteen routinely recorded clinical attributes. The system is built on the UCI Heart Disease dataset and uses a Logistic Regression classifier, chosen for its transparency, low computational cost and suitability as a baseline. Imputation, scaling and encoding are bundled with the classifier in a single Scikit-learn pipeline, which is fitted on the training partition only so that no information from the test partition leaks into model fitting. The fitted pipeline is serialised with Joblib and served through a Flask application that offers a prediction form, a results page and an analytical dashboard. Model quality is judged with accuracy, precision, recall, F1-score and a confusion matrix, with particular attention to false negatives, which are the costliest error in a screening setting. The paper describes the dataset, architecture, methodology, evaluation protocol, test plan and limitations of the system. The tool is intended for education and research and is not a substitute for medical examination or diagnosis*

Key Words: heart disease prediction; machine learning; logistic regression; Flask; clinical decision support; UCI dataset; classification metrics

1. INTRODUCTION

Cardiovascular conditions are a major source of illness and death across the globe, and the global burden analyses published over the past decade show that the problem continues to grow with ageing and urbanising populations [1], [2]. Because the disease frequently develops silently, the interval between the first measurable abnormality and a serious cardiac event is a valuable window for intervention. Clinicians use a combination of history taking, physical examination, electrocardiography and exercise testing to judge a patient's condition, and no software tool should be mistaken for a replacement for that process.

At the same time, hospitals and research groups have accumulated large collections of structured clinical records. Machine learning offers a way to examine such records systematically and to learn statistical relationships between measurable attributes, such as age, blood pressure, cholesterol and exercise response, and a documented outcome. A model of this kind can act as a teaching aid, a research baseline or a prototype for future decision-support work [3].

Many student and research prototypes, however, stop at a notebook that prints a single accuracy value. They rarely offer an interface that non-programmers can use, a clear view of how the model behaves on different kinds of errors, or an honest discussion of limitations. This work addresses that gap by describing a complete, reproducible workflow in which a Logistic Regression model is trained on a public dataset, evaluated with several complementary metrics, and delivered through a Flask web application with an integrated dashboard.

The main contributions of the paper are as follows:

• a leakage-aware training workflow in which every data-dependent transformation lives inside one pipeline object that is fitted on training data only;

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

• a modular architecture that separates offline training from online prediction and connects them through a serialised model artefact;
• an evaluation protocol that reports accuracy, precision, recall, F1-score and a confusion matrix, and discusses the clinical meaning of each error type; and
• a test plan and a frank list of limitations that frame the system as an educational demonstrator rather than a diagnostic device.

The remainder of the paper is organised as follows. Section II reviews background and related work, Section III describes the dataset, Section IV presents the system architecture, Section V explains the methodology, Section VI summarises the classification algorithm, Section VII sets out the evaluation framework, Section VIII covers implementation and testing, Section IX discusses limitations and responsible use, and Sections X and XI give future directions and conclusions.

II. BACKGROUND AND RELATED WORK

The Cleveland Clinic subset of the heart disease collection became a widely used benchmark after Detrano and colleagues reported a probability-based method for estimating the likelihood of angiographically confirmed coronary artery disease across several international centres [4]. Since then, the processed version of the data has been used in a very large number of studies that compare classifiers such as decision trees, k-nearest neighbours, support vector machines, naive Bayes, random forests and neural networks. Because the sample is small, results obtained on it vary considerably with the choice of data split, the handling of missing values and the metric reported, which is why careful experimental design matters more than the particular algorithm.

Logistic regression has a long history in medical statistics, where it is valued because each coefficient can be read as a change in log-odds and therefore explained to a clinician [5]. In modern libraries it is also a convenient baseline: it trains in milliseconds on tabular data, exposes calibrated-looking probabilities and rarely overfits badly when regularised. More flexible models may achieve higher scores on some splits, but they are harder to interpret and need more data to justify their added complexity. For an educational system whose purpose is to demonstrate a complete workflow, a transparent baseline is therefore a sound first choice.

A separate line of work stresses that model code and deployment practice deserve as much attention as accuracy. A common pitfall is data leakage, where information from the evaluation data influences preprocessing or model selection and inflates the reported performance [6]. Pipeline objects in Scikit-learn [7] were designed to reduce this risk, and the present work adopts them from the outset.

III. DATASET DESCRIPTION

The experiments are based on the Heart Disease dataset hosted by the UCI Machine Learning Repository [8]. The repository combines data collected at four sites: Cleveland, Hungary, Switzerland and Long Beach. The original database stores 76 attributes, but published studies, including this one, work with a subset of 14 of them. This paper assumes the commonly used processed Cleveland file, which holds 303 patient records, thirteen predictor attributes and one outcome attribute. A handful of records in this file have missing entries for the number of major vessels and the thalassemia category, and these are handled during preprocessing.

The outcome attribute records the angiographic finding on an integer scale from 0 to 4, where 0 denotes no significant narrowing and values from 1 to 4 denote increasing disease. In most classification studies the scale is collapsed into a binary label, with 0 meaning absence and any positive value meaning presence. The exact coding of the selected file is verified against the repository documentation before training, and the mapping is recorded in the final project report. Table I lists the predictors.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

TABLE I: CLINICAL ATTRIBUTES USED AS MODEL INPUTS

Attribute	Type	Unit / values	Description
age	Numeric	years	Age of the patient at examination
sex	Binary	0 / 1	Biological sex recorded in the file
cp	Categorical	1–4	Chest pain type: typical angina, atypical angina, non-anginal pain, asymptomatic
trestbps	Numeric	mm Hg	Resting systolic blood pressure on admission
chol	Numeric	mg/dl	Serum cholesterol
fbs	Binary	0 / 1	Fasting blood sugar above 120 mg/dl
restecg	Categorical	0–2	Resting electrocardiographic result
thalach	Numeric	beats/min	Maximum heart rate achieved during exercise
exang	Binary	0 / 1	Exercise-induced angina
oldpeak	Numeric	mm	ST depression induced by exercise relative to rest
slope	Categorical	1–3	Slope of the peak-exercise ST segment
ca	Ordinal	0–3	Number of major vessels coloured by fluoroscopy
thal	Categorical	3, 6, 7	Thalassemia-related category: normal, fixed defect, reversible defect

IV. SYSTEM ARCHITECTURE

The application is organised around a strict separation between an offline phase, in which the model is created, and an online phase, in which the stored model answers user requests. Fig. 1 shows both phases. During the offline phase the dataset is loaded, inspected, cleaned and split; a preprocessing-plus-classifier pipeline is fitted on the training partition and scored on the held-out partition. The fitted pipeline is then written to disk as a single Joblib file. During the online phase the Flask server loads that file once at start-up and reuses it for every request, so users never wait for training and the exact transformations applied at training time are reapplied to new input.

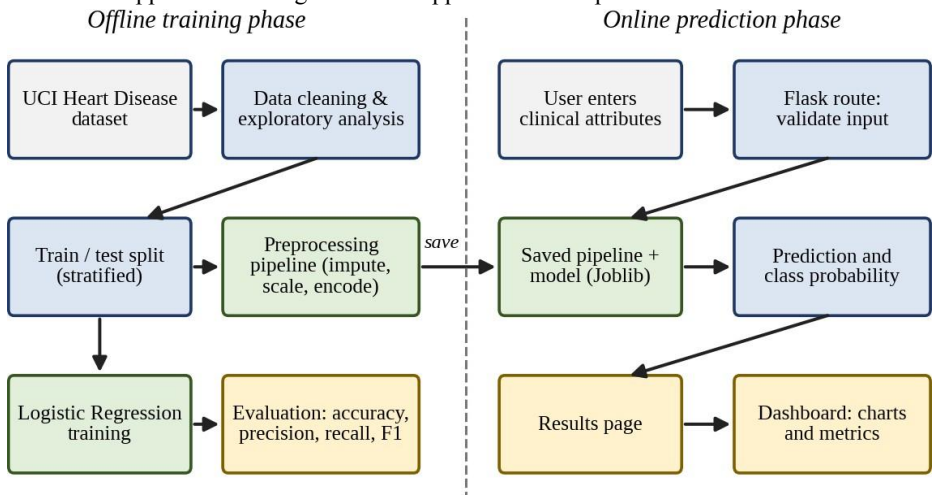


Fig. 1. Overall architecture of the proposed system, separating the offline training phase from the online prediction phase.

The system is divided into seven cooperating modules.

- Dataset management loads the file, checks its structure and confirms the meaning of the target variable.
- Preprocessing treats missing values and prepares numeric and categorical columns for the model.
- Model training fits the Logistic Regression classifier inside the shared pipeline.
- Model evaluation computes classification metrics and the confusion matrix on unseen data.
- Prediction accepts validated user input and returns a class label with its estimated probability.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

• Web interface supplies the home page, prediction form, results page and dashboard.
• Visualisation renders distribution charts and evaluation summaries for the dashboard.

The user-facing part of the application consists of four pages connected as shown in Fig. 2. The home page introduces the project and states its educational purpose. The form page collects the thirteen attributes with sensible ranges and short hints. The results page displays the predicted class together with the model's probability estimate and a reminder that the output is not a diagnosis. The dashboard presents exploratory charts and evaluation metrics so that users can judge how far the model can be trusted. The interface is built with Bootstrap so that it adapts to phone, tablet and desktop screens.

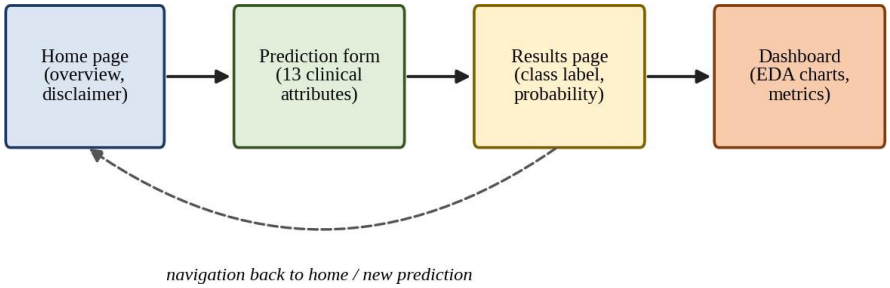


Fig. 2. Page flow of the Flask web application.

V. METHODOLOGY

The work follows ten sequential steps, summarised in Fig. 3. Although the steps are listed in order, in practice the exploratory and preprocessing stages are revisited several times as the data are better understood.

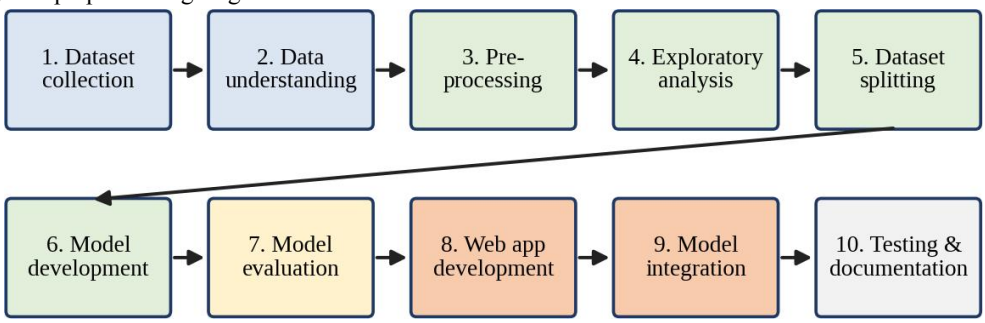


Fig. 3. The ten-step methodology followed in this work, from dataset collection to testing and documentation.

A. Data understanding and preprocessing

The first task is to inspect the shape of the table, the data type of each column, the range of each numeric attribute, the number of missing values and the balance between the two outcome classes. Missing entries are filled by an imputer whose statistics (the median for numeric columns and the most frequent category for categorical columns) are learned from the training partition only. Binary and ordinal columns are passed through unchanged, nominal columns such as chest pain type are one-hot encoded, and continuous columns are standardised to zero mean and unit variance. Standardisation is important for Logistic Regression because the regularised optimiser is sensitive to feature scale.

B. Exploratory data analysis

Histograms and box plots show how age, resting blood pressure, cholesterol, maximum heart rate and ST depression are distributed, and bar charts display the class balance and the relationship between categorical attributes such as chest pain type and the outcome. A correlation heat map highlights pairs of numeric attributes that move together. These charts guide preprocessing decisions, for example whether a variable needs transformation, and they are reused in the dashboard.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

C. Dataset splitting and leakage control

The records are divided into a training set and a test set with a stratified split, so that both partitions preserve the class proportions of the full data, and a fixed random seed is used so that the experiment can be repeated exactly. The test set is not touched while preprocessing choices are made or while hyper-parameters are tuned. Where tuning is needed, cross-validation is performed inside the training partition. Because the imputer, scaler and encoder are members of the same pipeline as the classifier, they are re-fitted on the training folds each time, which prevents statistics from the evaluation data from leaking into the model [6], [7].

D. Model development and persistence

A column transformer applies the appropriate treatment to each group of attributes, and its output feeds the Logistic Regression estimator, giving one object that accepts raw attribute values and returns a class and probability. After the evaluation step is complete, the pipeline is serialised with Joblib. The Flask application loads this file at start-up, so the same preprocessing logic is applied to training data and to every web request.

VI. CLASSIFICATION ALGORITHM

Logistic Regression is a supervised linear model for binary outcomes. For an input vector x of preprocessed attributes, the model first forms a weighted sum and then converts it into a probability with the logistic (sigmoid) function:

$$z = wTx + b \quad (1)$$

$$P(y = 1 \mid x) = \sigma(z) = 1 / (1 + e^{-*z*}) \quad (2)$$

Here w is the vector of learned coefficients and b is the intercept. The curve in Fig. 4 shows how the sigmoid maps any real-valued score to the interval between 0 and 1. A new record is assigned to the positive class (disease present) when its probability exceeds a decision threshold, conventionally 0.5. Training chooses w and b to minimise the regularised log-loss over the n training examples:

$$J(w, b) = -(1/n) \Sigma [\, y(i) \log p(i) + (1 - y(i)) \log(1 - p(i)) \,] + (1/2C) \; \| w \| 2 \quad (3)$$

The final term is an L2 penalty whose strength is governed by the inverse-regularisation parameter C; smaller values of C produce smaller coefficients and a simpler decision boundary. The optimisation problem is convex, so a standard solver reaches a global optimum without sensitivity to the starting point.

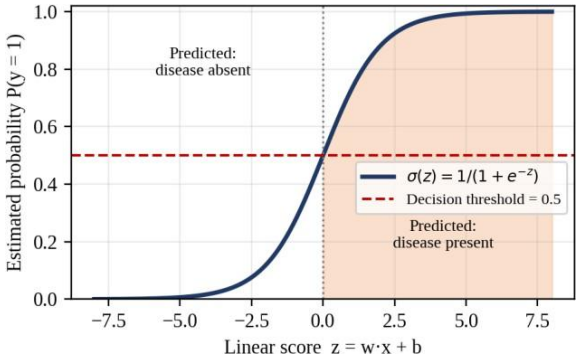


Fig. 4. The logistic function maps a linear score to a probability; a threshold of 0.5 separates the two predicted classes.

The algorithm suits this project for three reasons. First, its coefficients can be inspected and explained, which supports the educational aim. Second, it is inexpensive to train and to evaluate, so the web application responds immediately. Third, it provides a defensible baseline against which more complex models can be compared in later work. Its main drawback is that it assumes a linear relationship between the attributes and the log-odds of the outcome, so it can miss interactions unless they are engineered explicitly.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

VII. EVALUATION FRAMEWORK

Accuracy alone can be misleading, particularly when the classes are unequal in size or when errors have unequal consequences. The model is therefore judged with several complementary measures derived from the confusion matrix in Fig. 5. Treating disease as the positive class, a true positive is a diseased case that is correctly flagged, a false negative is a diseased case that the model misses, a false positive is a healthy case that is wrongly flagged, and a true negative is a healthy case that is correctly cleared [9].

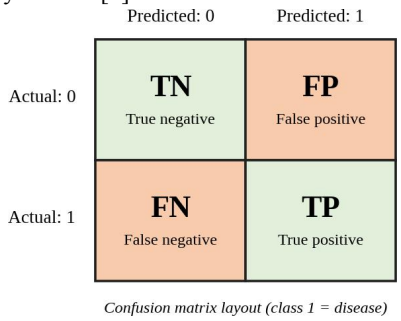


Fig. 5. Layout of the binary confusion matrix used to derive all evaluation metrics.

TABLE II: EVALUATION METRICS AND THEIR INTERPRETATION

Metric	Formula	Interpretation in this project
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall share of correct predictions; can hide poor performance on the smaller class
Precision	TP / (TP + FP)	Of the cases flagged as diseased, the share that truly are; low precision means many false alarms
Recall	TP / (TP + FN)	Of the truly diseased cases, the share detected; the most important figure in a screening context
F1-score	2 · Precision · Recall / (Precision + Recall)	Harmonic mean that balances false alarms against missed cases

Because a missed case is usually more harmful than a false alarm in a screening scenario, recall receives special attention, and the decision threshold can be lowered from 0.5 to trade some precision for fewer missed cases. Results will be reported on the held-out test partition and, where the sample allows, as the mean and spread across repeated stratified cross-validation folds, since a single split of roughly three hundred records produces unstable estimates. No performance figure is assumed in advance; the layout in Table III is used to record the measured values once the experiments have been run.

VIII. IMPLEMENTATION AND TESTING

A. Software environment

The system is implemented in Python. Pandas [10] and NumPy [11] handle tabular data and numerical operations, Scikit-learn [7] supplies the preprocessing components, the Logistic Regression estimator and the metric functions, and Matplotlib [12] or Plotly produce the charts. Joblib stores the fitted pipeline, and Flask [13] provides routing and page templates, with HTML, CSS, Bootstrap and JavaScript used on the client side. A standard laptop with a dual-core processor and 4 GB of memory is sufficient, and the project is managed with Git so that the code and the results can be reproduced.

B. Test plan

Testing covers both the machine learning components and the web application. The cases in Table IV are designed to confirm that valid input produces a sensible result, that invalid input is rejected with a clear message rather than a server error, and that the pages remain usable on small screens.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 1, November 2026
Impact Factor: 8.2

TABLE IV: PLANNED TEST CASES FOR THE WEB APPLICATION

No.	Test case	Expected behaviour	Area
1	All fields supplied within valid ranges	Prediction and probability displayed	Prediction
2	A required field left empty	Form rejected with a field-specific message	Validation
3	Non-numeric text in a numeric field	Input refused; no server error	Validation
4	Value outside plausible clinical range	Warning shown; prediction blocked	Validation
5	Boundary values at range limits	Accepted and processed consistently	Preprocessing
6	Same input submitted twice	Identical output both times	Reproducibility
7	Pages opened on phone, tablet, desktop	Layout adapts; no horizontal scrolling	Responsiveness
8	Dashboard opened	Charts and metrics load correctly	Visualisation

IX. LIMITATIONS AND RESPONSIBLE USE

Several limitations should be kept in mind when interpreting the system's output. The processed dataset is small and was collected at specific centres and times, so the learned patterns may not carry over to other populations, age groups or healthcare settings. Records contain missing or inconsistently recorded values, and the imputation strategy, however sensible, cannot recover information that was never measured. A linear model cannot represent every interaction among attributes, and the thirteen predictors are only a fraction of the information that clinicians consider. Finally, no external clinical validation is within the scope of this academic project.

For these reasons, the prediction produced by the application does not establish the presence or absence of disease in any individual. The results page and home page state this plainly, and the application does not store personal data entered by users. Any future clinical use would need independent validation, expert oversight and compliance with privacy and healthcare regulations.

X. FUTURE SCOPE

The project offers several directions for extension. Other algorithms, such as regularised tree ensembles and gradient boosting, can be compared with the baseline under the same evaluation protocol. Coefficient inspection, permutation importance and similar techniques can improve interpretability. Testing on independent datasets would reveal how well the model generalises, and probability calibration together with a study of decision thresholds would clarify the consequences of false-negative predictions. On the engineering side, the dashboard can be enriched with additional statistical views, and secure user accounts with prediction history could be added if privacy safeguards are put in place. Larger and more diverse data, together with a review of deployment and data-protection practices, are prerequisites for any step beyond demonstration.

XI. CONCLUSION

This paper described the design of an end-to-end machine learning application for heart disease risk classification. A Logistic Regression model, trained on the public UCI Heart Disease dataset inside a leakage-aware preprocessing pipeline, is delivered through a Flask web interface with a prediction form, a results page and an analytical dashboard. A multi-metric evaluation protocol, a structured test plan and an explicit discussion of limitations frame the work as a transparent, reproducible educational demonstrator. The measured performance will depend on the dataset version, the split and the preprocessing choices, and it must be read together with the stated limitations. The project provides a practical foundation on which further research into predictive modelling and healthcare analytics can be built.

REFERENCES

[1]. G. A. Roth et al., “Global burden of cardiovascular diseases and risk factors, 1990–2019: Update from the GBD 2019 study,” J. Am. Coll. Cardiol., vol. 76, no. 25, pp. 2982–3021, Dec. 2020.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 1, November 2026 Impact Factor: 8.2

[2]. World Health Organization, "Cardiovascular diseases (CVDs)." [Online]. Available: https://www.who.int/health-topics/cardiovascular-diseases
[3]. D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, Applied Logistic Regression, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.
[4]. R. Detrano et al., "International application of a new probability algorithm for the diagnosis of coronary artery disease," Am. J. Cardiol., vol. 64, no. 5, pp. 304–310, Aug. 1989.
[5]. Scikit-learn Developers, "Logistic regression," Scikit-learn documentation. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
[6]. S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," ACM Trans. Knowl. Discov. Data, vol. 6, no. 4, art. 15, Dec. 2012.
[7]. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html
[8]. UCI Machine Learning Repository, "Heart Disease Dataset," University of California, Irvine. [Online]. Available: https://archive.ics.uci.edu/dataset/45/heart+disease
[9]. M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Inf. Process. Manag., vol. 45, no. 4, pp. 427–437, Jul. 2009.
[10]. W. McKinney, "Data structures for statistical computing in Python," in Proc. 9th Python in Science Conf., 2010, pp. 56–61.
[11]. C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, Sep. 2020.
[12]. J. D. Hunter, "Matplotlib: A 2D graphics environment," Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, 2007.
[13]. Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com/