# Final Project Report

**Heart Disease Prediction System using Machine Learning**

Team ID: NHA-011

Supervisor: Aya Abdullah

Team Members:

- Ahmed Mostafa Ahmed
- Seif Eldeen Gamal
- Arwa Elwa
- Mohamed Ayoub
- Logine Rashed
- Habiba Ayman

Year: 2025

## Abstract

Heart disease is one of the leading causes of death worldwide. Early detection of patients at high risk can significantly improve treatment outcomes and reduce mortality rates. This project presents a machine learning–based Heart Disease Prediction System that predicts whether a patient is likely to have heart disease based on clinical features such as age, blood pressure, cholesterol level, chest pain type, and other medical attributes.

The project follows the full lifecycle of a predictive analytics solution: data collection and preprocessing, exploratory data analysis, model development and optimization, deployment through a backend API, and integrating a simple web user interface. The final model is exposed via an API and consumed by a web UI where the user can enter patient data and receive a real-time prediction (disease / no disease).

This report documents the dataset, methodology, implementation details, results, limitations, and possible future improvements. The system is designed for educational purposes and is not intended for real clinical use.

## 1. Introduction

Cardiovascular diseases (CVDs) represent a major global health problem, causing millions of deaths each year. Detecting patients at high risk of heart disease early can help doctors intervene sooner and reduce the likelihood of severe complications. Traditional diagnosis is based on medical expertise, tests, and clinical guidelines, which can be time-consuming and resource-intensive.

With the availability of clinical datasets and electronic health records, machine learning can support decision-making by learning patterns from historical patient data. These models can provide a risk estimation for heart disease based on multiple clinical attributes.

In this project, we develop a Heart Disease Prediction System that trains a classification model on a labeled heart disease dataset, evaluates different algorithms, selects the best-performing model, and deploys it in a simple web application. The goal is to demonstrate how data science and machine learning can be applied to a real healthcare-related problem.

## 2. Project Objectives

The main objectives of this project are:

- To understand and preprocess a heart disease dataset, including handling missing values and encoding features.
- To perform exploratory data analysis (EDA) and identify patterns and risk factors related to heart disease.
- To implement and compare several machine learning algorithms for binary classification.
- To evaluate models using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
- To select and save the best model as a serialized file that can be loaded later for inference.
- To develop a backend API that serves predictions using the trained model.
- To implement a simple user interface (UI) that communicates with the API and displays predictions.
- To log predictions for basic monitoring and potential future re-training.

## 3. Dataset Description

The dataset used in this project is a structured heart disease dataset in tabular form. Each row corresponds to a patient, and each column corresponds to a clinical feature or the diagnosis label. The dataset is commonly used in research and education for classification tasks.

### 3.1 Features

- age: Patient's age in years.
- sex: Gender (e.g., 0 = female, 1 = male).
- cp: Chest pain type (categorical).
- trestbps: Resting blood pressure (mm Hg).
- chol: Serum cholesterol (mg/dl).
- fbs: Fasting blood sugar (> 120 mg/dl).
- restecg: Resting electrocardiographic results.
- thalach: Maximum heart rate achieved.
- exang: Exercise-induced angina.
- oldpeak: ST depression induced by exercise relative to rest.
- slope: Slope of the peak exercise ST segment.
- ca: Number of major vessels colored by fluoroscopy.
- thal: Thalassemia (categorical).

### 3.2 Target Variable

The target variable is a binary label indicating the presence of heart disease:

- 0 → No heart disease
- 1 → Presence of heart disease

### 3.3 Train–Test Split

The dataset is split into training and testing subsets to estimate how well the model generalizes to unseen data. Typically, 70% of the data is used for training and 30% for testing, or a similar ratio depending on the size of the dataset.

## 4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was performed in the Jupyter Notebook (final_depi_mil4.ipynb). EDA helps in understanding feature distributions, detecting outliers and missing values, and examining relationships between features and the target variable.

### 4.1 Class Distribution

The distribution of the target classes (patients with and without heart disease) was analyzed to determine if the dataset is balanced. If one class dominates, class imbalance techniques such as resampling or class weighting may be required.

### 4.2 Univariate Analysis

Individual features were visualized using histograms and bar plots to inspect their ranges, central tendencies, and variability. This step helps identify unusual values and skewed distributions.

### 4.3 Bivariate Analysis

The relationship between features and the target variable was explored. For example, we compared age, cholesterol, and maximum heart rate distributions for patients with and without heart disease. Categorical features such as chest pain type and thalassemia were also analyzed across both classes.

## 5. Data Preprocessing & Feature Engineering

Before training the machine learning models, the data was cleaned and transformed to a suitable format. The main preprocessing and feature engineering steps included:

- Handling missing values with appropriate imputation strategies or row removal.
- Encoding categorical variables (e.g., chest pain type, thalassemia, slope) into numerical form.
- Feature scaling for algorithms that are sensitive to the magnitude of input values.
- Splitting the data into training and test sets.

### 5.1 Handling Missing Values

Missing values were checked for each feature. Depending on the proportion and importance of the missing data, rows were either removed or missing values were imputed using mean, median, or mode.

### 5.2 Encoding Categorical Features

Categorical features such as chest pain type, rest ECG, slope, and thal were encoded into numerical values using label encoding or one-hot encoding, depending on the model requirements.

### 5.3 Feature Scaling

For models that are sensitive to the scale of the input data, such as Logistic Regression or Support Vector Machines, numerical features were standardized or normalized. Tree-based models may not require scaling.

## 6. Model Development and Optimization

Multiple machine learning models were trained and compared to select the best-performing classifier for heart disease prediction. Example models include Logistic Regression, Decision Trees, Random Forests, and XGBoost.

### 6.1 Training Procedure

- Train each candidate model on the training set.
- Evaluate the models on the test set using appropriate metrics.
- Compare the results and select the best model based on performance.
- Optimize hyperparameters of the best model to improve accuracy and generalization.
- Save the final optimized model as best_model_optimized.pkl.

### 6.2 Evaluation Metrics

- Accuracy: Overall percentage of correct predictions.
- Precision: Percentage of predicted positives that are actually positive.
- Recall (Sensitivity): Percentage of actual positives detected correctly.
- F1-score: Harmonic mean of precision and recall.
- Confusion matrix: Breakdown of true positives, true negatives, false positives, and false negatives.

### 6.3 Hyperparameter Tuning

Hyperparameter tuning was performed using techniques such as grid search or random search to find better combinations of parameters. The final chosen model achieved a good balance between accuracy and recall, which is important in medical screening tasks.

## 7. System Implementation

The final system consists of two main parts: a backend API responsible for serving the model predictions and a frontend user interface that allows users to enter patient data and view the prediction.

### 7.1 Backend API (app.py)

The backend is implemented in app.py. Its responsibilities include loading the trained model file best_model_optimized.pkl, receiving HTTP requests that contain patient data, preprocessing the data in the same way as during training, and returning the model's prediction as a response.

### 7.2 Frontend UI (ui.py)

The frontend user interface is implemented in ui.py. It provides input fields for all relevant clinical features and a button to trigger prediction. Upon submission, the UI sends the data to the backend API and displays the prediction result as either likely heart disease or no heart disease.

### 7.3 Logging and Monitoring

Each prediction request, including the input features and corresponding model output, can be logged into prediction_logs.csv. These logs can be analyzed later for model monitoring, auditing, or to support future retraining of the model.

## 8. Results and Discussion

After experimenting with different models, the final selected model provided satisfactory performance on the test set. The evaluation metrics indicate that the model can correctly classify a high proportion of patients, with a reasonable trade-off between precision and recall.

In a medical context, recall (sensitivity) is particularly important because missing a true positive case (a patient who actually has heart disease) can be dangerous. Therefore, priority was given to models that achieved good recall while maintaining acceptable precision and overall accuracy.

Actual numerical values for accuracy, precision, recall, and F1-score can be filled in here based on the final training notebook results (e.g., Accuracy: XX.X%, Recall: YY.Y%, etc.).

## 9. Deployment and MLOps Aspects

From an MLOps perspective, the project separates model training from model inference. The training logic is contained in the Jupyter Notebook, while inference is handled by the app.py API. The trained model is saved as a file and loaded only when needed, avoiding retraining at prediction time.

The project is version-controlled using Git and hosted on GitHub, making it easier to track changes and collaborate. Basic logging through prediction_logs.csv provides simple monitoring that could be extended in future work.

## 10. Conclusion

This project demonstrates how machine learning techniques can be applied to predict heart disease risk using a clinical dataset. The workflow covers data preprocessing, exploratory analysis, model development, evaluation, and deployment in the form of an API and a user-friendly interface.

While the system shows promising results, it should be considered an educational prototype rather than a certified medical tool. Further validation on larger and more diverse datasets, along with expert medical review, would be required before considering real-world deployment.