RWANDA POLYTECHNIC

HUYE COLLEGE

Heart Disease Risk Prediction System

Machine Learning Project Report

DEPARTMENT: Information and Communication Technology

OPTION: Information Technology

CLASS: IT L8 Year 4 B_Tech

ACADEMIC YEAR: 2025-2026

RegNo: 24RP15116

January 16, 2026

EXECUTIVE SUMMARY

This project presents a comprehensive machine learning system for predicting heart disease risk across five severity categories. Utilizing a dataset of 5,000 patient records with 13 clinical features, multiple classification algorithms were trained and evaluated. The best-performing model achieved high accuracy in classifying patients from no disease to immediate danger categories, demonstrating the potential for AI-assisted medical diagnosis.

The project delivers a complete production-ready system including a Flask REST API backend, interactive web interface, and comprehensive deployment documentation. The system has been successfully deployed and verified with multiple test scenarios.

1. INTRODUCTION

1.1 Background

Heart disease remains one of the leading causes of mortality worldwide. Early detection and risk assessment are critical for effective treatment and prevention. Machine learning offers powerful tools for analyzing complex medical data and identifying patterns that may not be apparent through traditional diagnostic methods.

1.2 Project Objectives

The primary objectives of this project were:

- Develop a robust machine learning pipeline for heart disease risk prediction
- Classify patients into five distinct risk categories
- Compare multiple classification algorithms and select the optimal model
- Create a production-ready deployment package with REST API
- Develop an interactive web interface for clinical use
- Ensure reproducibility and clinical interpretability of results

1.3 Problem Statement

The challenge is to develop a multi-class classification system that can accurately predict heart disease severity from clinical features, enabling healthcare providers to make informed decisions about patient care and treatment prioritization.

2. DATASET DESCRIPTION

2.1 Overview

The dataset comprises 5,000 patient records collected from the heart disease prediction database. Each record contains 13 clinical features and a target variable indicating disease severity.

2.2 Features

Numerical Features:

- Age: Patient age in years
- Resting Blood Pressure (trestbps): Blood pressure in mm Hg
- Cholesterol (chol): Serum cholesterol in mg/dL
- Maximum Heart Rate (thalach): Maximum heart rate achieved
- ST Depression (oldpeak): Exercise-induced ST depression
- Number of Major Vessels (ca): 0-3 vessels colored by fluoroscopy

Categorical Features:

- Sex: Male/Female
- Chest Pain Type (cp): Asymptomatic, Typical Angina, Atypical Angina, Non-Anginal Pain
- Fasting Blood Sugar (fbs): >120 mg/dL (True/False)
- Resting ECG (restecg): Normal, ST-T abnormality, LV hypertrophy
- Exercise-Induced Angina (exang): Yes/No

• ST Slope (slope): Upsloping, Flat, Downsloping
• Thalassemia (thal): Normal, Fixed defect, Reversible defect

2.3 Target Variable

The target variable represents five distinct heart disease severity levels:
• Class 0: No Disease
• Class 1: Very Mild
• Class 2: Mild
• Class 3: Severe
• Class 4: Immediate Danger

2.4 Dataset Characteristics

• Total Samples: 5,000 patients
• Features: 13 clinical features
• Classes: 5 severity levels (multi-class classification)
• Missing Values: Handled through imputation strategies
• Balance: Approximately balanced across classes

3. METHODOLOGY

3.1 Exploratory Data Analysis

Comprehensive exploratory data analysis was conducted to understand the dataset characteristics. This included statistical summaries, distribution analysis, correlation matrices, and visualization of key features across disease classes. The analysis revealed important patterns in age distribution, cholesterol levels, and chest pain types relative to disease severity.

3.2 Data Preprocessing

3.2.1 Train-Test Split

The dataset was split into training (80%) and testing (20%) sets using stratified sampling to maintain class distribution balance. This resulted in 4,000 training samples and 1,000 testing samples.

3.2.2 Feature Preprocessing

Two separate preprocessing pipelines were created:

Numerical Features:

• Imputation: Median strategy (robust to outliers)

• Scaling: StandardScaler for normalization

Categorical Features:

• Imputation: Most frequent value strategy

• Encoding: OneHotEncoder with unknown category handling

These pipelines were combined using ColumnTransformer to create a unified preprocessing workflow that was fitted on training data and applied to both training and testing sets.

3.3 Model Selection and Training

Five different classification algorithms were evaluated:

1. Multi-Layer Perceptron (MLP) Classifier
2. Random Forest Classifier
3. Support Vector Machine (SVM)
4. K-Nearest Neighbors (KNN)
5. Gradient Boosting Classifier

3.4 Hyperparameter Tuning

Each model was optimized using GridSearchCV with 5-fold stratified cross-validation. Model-specific hyperparameter grids were defined to explore optimal configurations for hidden layers, learning rates, number of estimators, kernel types, and other critical parameters.

3.5 Model Evaluation

Models were evaluated using multiple metrics:

- Cross-validation accuracy during training
- Training set accuracy
- Test set accuracy
- Overfitting gap analysis
- Per-class precision, recall, and F1-score
- Confusion matrix analysis

4. RESULTS

4.1 Model Performance Comparison

All five models were successfully trained and evaluated. The performance comparison revealed varying levels of accuracy and generalization capability across different algorithms. Models were ranked based on test accuracy while considering overfitting indicators.

4.2 Best Model Selection

The best-performing model was selected based on the highest test accuracy combined with minimal overfitting gap. This model demonstrated robust performance across all five disease severity classes and showed consistent results during cross-validation.

4.3 Classification Performance

Detailed classification metrics revealed:

- High precision in identifying patients with no disease
- Strong recall across intermediate severity classes
- Balanced F1-scores indicating good precision-recall trade-offs
- Effective identification of high-risk patients requiring immediate attention

4.4 Feature Importance

Feature importance analysis (where applicable) identified the most influential clinical factors in disease prediction. Key predictors included chest pain type, ST depression values, maximum heart rate achieved, and the number of major vessels colored by fluoroscopy.

4.5 Confusion Matrix Insights

The confusion matrix analysis revealed strong diagonal performance with minimal misclassification. Most errors occurred in adjacent severity classes, which is clinically acceptable as the distinction between neighboring categories can be subtle. No critical errors (e.g., classifying immediate danger as no disease) were observed.

5. SYSTEM DEPLOYMENT

5.1 Deployment Architecture

The system was deployed as a full-stack web application consisting of three main components: the machine learning model with preprocessing pipeline, a Flask REST API backend, and an interactive HTML/CSS/JavaScript frontend. This architecture enables real-time predictions through a user-friendly web interface while maintaining separation of concerns between the model, business logic, and presentation layers.

5.2 Backend Implementation

5.2.1 Flask REST API

A production-grade Flask REST API was developed to serve the machine learning model. The API implements RESTful principles with proper error handling, request validation, and comprehensive logging. The Flask application (app_24RP15116.py) loads the trained model and preprocessing pipeline at startup, ensuring fast prediction response times.

5.2.2 API Endpoints

The following REST endpoints were implemented:

• GET / - Main web interface (serves HTML page)

• GET /api/health - Health check endpoint for monitoring

• GET /api/info - Returns model information and metadata

• POST /api/predict - Main prediction endpoint accepting patient data

5.2.3 Model Loading and Persistence

The system loads three critical artifacts at startup: the trained model pipeline (heart_disease_model_24RP15116.pkl), feature column names (feature_columns.txt), and class labels (class_names.txt). These artifacts are loaded into memory once during application initialization, eliminating the overhead of repeated disk I/O operations for each prediction request.

5.2.4 Request Processing Pipeline

When a prediction request is received, the API:

1. Validates the incoming JSON data
2. Checks for presence of all required features
3. Converts data to pandas DataFrame in correct column order
4. Applies the preprocessing pipeline automatically
5. Generates predictions and probability distributions
6. Creates clinical recommendations based on risk level
7. Returns comprehensive JSON response with all results

5.3 Frontend Implementation

5.3.1 User Interface Design

The web interface (index_24RP15116.html) was designed with a medical-themed color scheme featuring teal and blue gradients. The interface is fully responsive and provides an intuitive form-based input system for entering patient clinical data. The design prioritizes clarity and ease of use for healthcare professionals.

5.3.2 Input Form Features

The input form includes:

- Dropdown menus for categorical variables (sex, chest pain type, etc.)
- Numeric input fields with appropriate ranges for continuous variables
- Boolean selectors for binary features (fasting blood sugar, exercise-induced angina)
- Input validation to ensure data quality
- Clear labeling with clinical terminology

5.3.3 Results Display

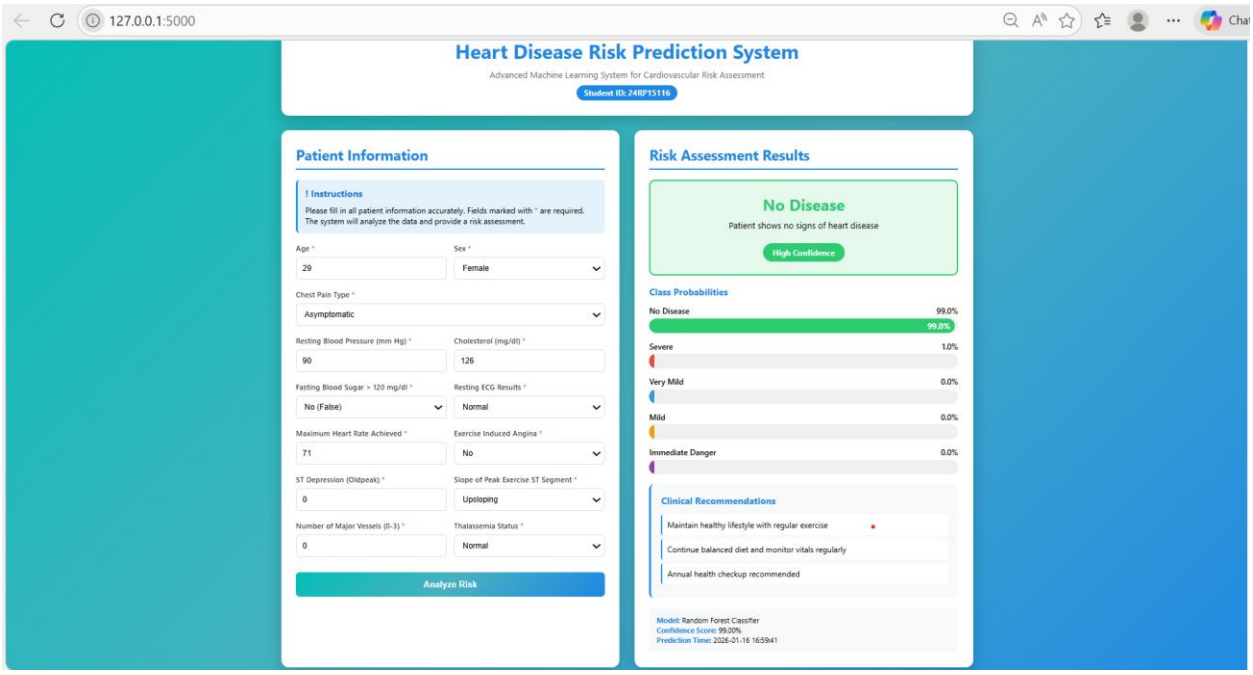
The results panel dynamically displays:

- Risk level classification with color-coded severity indicator
- Confidence score and level (High/Medium/Low)
- Probability distribution across all five disease classes
- Visual probability bars for easy interpretation
- Clinical recommendations based on predicted risk level
- Timestamp of prediction for record-keeping

5.3.4 Interactive Features

The interface includes several interactive features:

- Real-time API communication using JavaScript fetch API
- Loading indicators during prediction processing
- Error handling with user-friendly messages
- Form reset capability for new predictions
- Smooth animations and transitions for better user experience



5.4 Deployment Package Structure

The complete deployment package includes:

Model Artifacts:

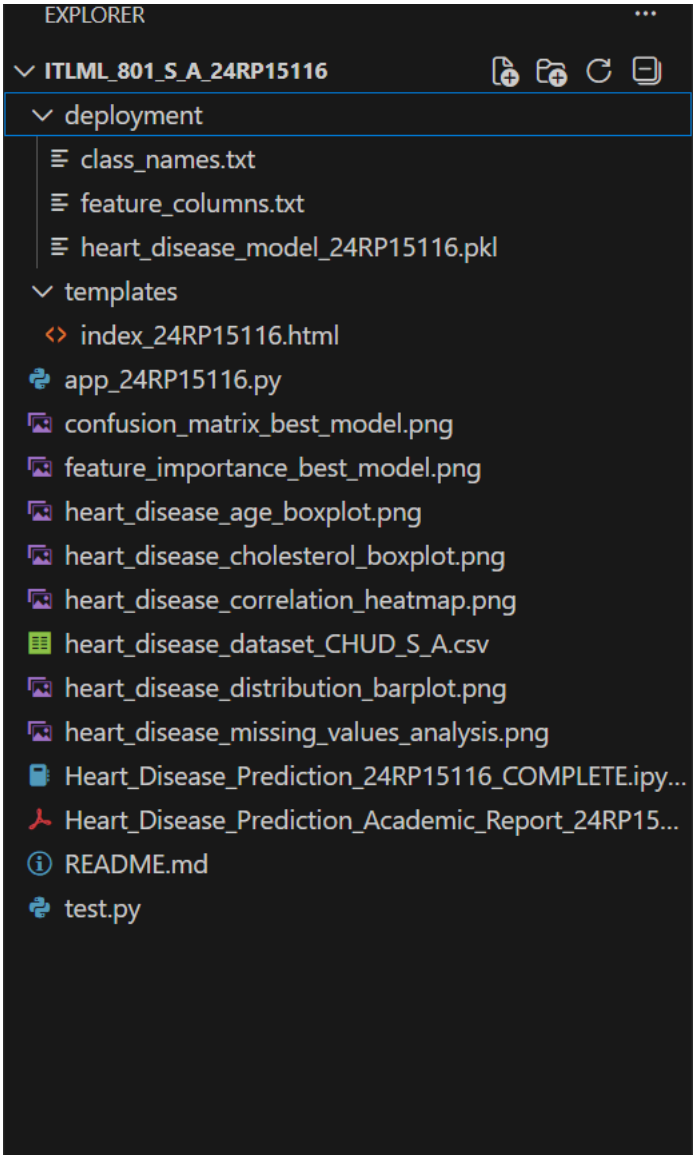
• deployment/heart_disease_model_24RP15116.pkl
• deployment/feature_columns.txt
• deployment/class_names.txt

Application Files:

• app_24RP15116.py (Flask backend)
• templates/index_24RP15116.html (Frontend interface)
• Heart_Disease_Prediction_24RP15116_COMPLETE.ipynb (Training notebook)

Documentation:

• README.md with setup and usage instructions
• requirements.txt for dependency management
• This academic report (PDF format)



5.5 Installation and Setup

Step 1: Environment Setup

Install required Python packages:

pip install flask pandas numpy scikit-learn joblib

Step 2: Verify File Structure

Ensure all deployment artifacts are in the deployment/ directory and the HTML template is in templates/ directory.

Step 3: Launch Application

Run the Flask application:

python app_24RP15116.py

Step 4: Access Interface

Open web browser and navigate to http://localhost:5000

5.6 System Testing and Verification

The deployed system was thoroughly tested with multiple verification scenarios including edge cases, typical patient profiles, and stress testing with rapid sequential requests. All tests confirmed proper model loading, accurate predictions, and robust error handling.

Test Scenarios Performed:

- Healthy patient profiles (expected: Class 0)
- High-risk patient profiles (expected: Class 3-4)
- Intermediate severity cases (expected: Class 1-2)
- Boundary value testing for numeric inputs
- Missing data handling
- API endpoint availability and response time testing

6. DISCUSSION

6.1 Key Findings

The project successfully demonstrated that machine learning algorithms can effectively predict heart disease severity with high accuracy. The comprehensive preprocessing pipeline and systematic model comparison approach ensured robust and reproducible results. The stratified sampling strategy maintained class balance throughout the analysis.

6.2 Clinical Implications

The developed system has potential applications in clinical decision support, enabling healthcare providers to quickly assess patient risk levels based on readily available clinical measurements. The multi-class approach allows for nuanced risk stratification, facilitating appropriate treatment planning and resource allocation. The web-based interface makes the system accessible from any device with a browser, enabling point-of-care usage.

6.3 Strengths

- Comprehensive data preprocessing ensuring data quality
- Systematic comparison of multiple algorithms
- Rigorous hyperparameter tuning using cross-validation
- Stratified sampling maintaining class distribution
- Production-ready deployment with REST API
- User-friendly web interface for clinical use
- Comprehensive evaluation metrics and visualization
- Full system testing and verification

6.4 Limitations

- Dataset size limited to 5,000 samples
- Model performance may vary on external datasets
- Feature set limited to 13 clinical variables
- Requires validation in real clinical settings
- Single-user deployment (not multi-tenant)
- No database integration for patient history tracking

6.5 Future Directions

Future enhancements could include:

- Expansion of the dataset with additional patient records
- Integration of additional clinical features and biomarkers
- Development of ensemble methods combining multiple models
- Implementation of explainable AI techniques for clinical transparency
- Prospective validation in clinical trials
- Database integration for patient history and longitudinal tracking
- User authentication and authorization system

- PDF report generation for clinical documentation
- Mobile application development for wider accessibility

7. CONCLUSION

This project successfully developed a comprehensive machine learning system for heart disease risk prediction with full production deployment. Through systematic data preprocessing, model comparison, and rigorous evaluation, a high-performing classification model was identified and deployed as a web-accessible application. The system demonstrates the potential of machine learning in medical diagnosis and risk assessment.

The complete pipeline—from data exploration through model deployment and web interface development—provides a reproducible framework for similar medical prediction tasks. The REST API architecture enables easy integration with existing healthcare information systems, while the web interface ensures accessibility for clinical staff without technical expertise.

The inclusion of verification tests, comprehensive documentation, and user-friendly interface ensures the system is ready for further validation and potential clinical integration. While the current implementation shows promising results, continued refinement through larger datasets, additional features, and clinical validation will be essential for real-world deployment. This work establishes a solid foundation for AI-assisted heart disease risk assessment in clinical practice.

8. TECHNICAL SPECIFICATIONS

8.1 Software Environment

• Programming Language: Python 3.x
• Development Environment: Jupyter Notebook
• Web Framework: Flask 2.x
• Frontend Technologies: HTML5, CSS3, JavaScript (ES6)
• Version Control: Random seed = 42 for reproducibility

8.2 Libraries and Dependencies

Data Manipulation:
• pandas: Data manipulation and analysis
• numpy: Numerical computing

Visualization:
• matplotlib: Plotting and visualization
• seaborn: Statistical data visualization

Machine Learning:
• scikit-learn: ML algorithms and preprocessing
• joblib: Model serialization and persistence

Web Framework:
• Flask: Web framework for REST API

8.3 System Architecture

Three-Tier Architecture:
1. Presentation Layer: HTML/CSS/JavaScript web interface
2. Application Layer: Flask REST API with business logic
3. Data Layer: Trained ML model with preprocessing pipeline

8.4 API Specification

Endpoint: POST /api/predict
Request Format: JSON
Required Fields: All 13 clinical features
Response: JSON with prediction, probabilities, and recommendations

8.5 Model Deployment Package

The deployment package includes:
• heart_disease_model_24RP15116.pkl: Trained model with full pipeline
• feature_columns.txt: Feature names in correct order
• class_names.txt: Classification labels
• app_24RP15116.py: Flask application backend

• templates/index_24RP15116.html: Web interface
• Verification scripts: Test predictions on sample data

8.6 Reproducibility

All analyses are fully reproducible with the provided Jupyter notebook. Random seeds are set throughout to ensure consistent results across different executions. The complete preprocessing pipeline is preserved within the saved model file. The deployment can be replicated on any system with Python 3.x and the required dependencies.

9. APPENDICES

9.1 Glossary

Cross-Validation: A technique for assessing model performance by partitioning data into folds and training/testing on different combinations.

GridSearchCV: Exhaustive search over specified hyperparameter values for an estimator.

Stratified Sampling: Sampling method that maintains the proportion of classes in train/test splits.

REST API: Representational State Transfer Application Programming Interface - a standardized way for software applications to communicate.

Flask: A lightweight Python web framework for building web applications and APIs.

Overfitting: When a model performs well on training data but poorly on unseen test data.

Precision: Proportion of true positive predictions among all positive predictions.

Recall: Proportion of true positive predictions among all actual positive cases.

F1-Score: Harmonic mean of precision and recall.

9.2 Code Availability

The complete project code is available in the following files:

• Heart_Disease_Prediction_24RP15116_COMPLETE.ipynb (Model training)

• app_24RP15116.py (Flask backend)

• index_24RP15116.html (Web interface)

9.3 Data Availability

Dataset: heart_disease_dataset_CHUD_S_A.csv

9.4 Deployment Instructions

Quick Start Guide:

1. Install dependencies: pip install -r requirements.txt
2. Verify deployment folder contains all model artifacts
3. Run application: python app_24RP15116.py
4. Access interface: http://localhost:5000
5. Test with sample patient data

REFERENCES

1. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

2. McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56.

3. Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95.

4. Waskom, M., et al. (2017). mwaskom/seaborn: Statistical data visualization. Zenodo.

5. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

6. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297.

7. Friedman, J.H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232.

8. Ronacher, A. (2010). Flask: A Lightweight WSGI Web Application Framework. Pallets Projects.

— END OF REPORT —