HEART DISEASE RISK PREDICTION PROJECT REPORT

A Machine Learning-Based Tool for Preventive Care Using the Cleveland Heart Disease Dataset

Contributed By

Omnia Arafah
Sandy Ashraf
Shahd Bakry
Sama Khaled
Sagdah Fathy
Ayatullah Ahmed

GTC Team

Contents

Chapter 1

Executive Summary

This report details the development of a machine learning-based tool for predicting heart disease risk using the Cleveland Heart Disease dataset. The project aims to classify individuals as high-risk or low-risk for heart disease, enabling preventive care. Key phases include data preparation, exploratory data analysis (EDA) and feature engineering, model training and validation, deployment via a web interface, and documentation.

The dataset contains 303 records with features like age, sex, chest pain type (cp), resting blood pressure (trestbps), cholesterol (chol), and more, with a binary target (0: low risk, 1: high risk). We leveraged Python libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn. Models evaluated include Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest, with Random Forest achieving the highest performance (accuracy $\sim$0.85, recall $\sim$0.90, F1-score $\sim$0.87 on test data). The final model is deployed as a simple web app for real-time predictions.

In this executive summary, we provide a high-level overview of the project's objectives, methodology, and outcomes. The subsequent sections delve deeper into each phase, explaining the processes, challenges, and insights gained.

Chapter 2

Introduction

2.1 Problem Statement

Heart disease is a leading cause of death worldwide, accounting for millions of fatalities annually. Early detection through risk prediction can enable preventive actions, such as lifestyle changes, medication, or medical interventions, potentially saving lives. Our healthcare team seeks to shift toward preventive care by building an ML tool that classifies patients as high-risk (1) or low-risk (0) based on clinical features. This allows clinicians to prioritize monitoring and interventions for those at greater risk.

We expand on this by noting that traditional diagnostic methods, like angiograms, are invasive and costly. An ML-based non-invasive predictor can serve as a preliminary screening tool, reducing the burden on healthcare systems.

2.2 Dataset Overview

- **Source:** Cleveland Heart Disease dataset (from UCI Machine Learning Repository, provided as "Heart_disease_cleveland_new.csv").
- **Size:** 303 rows, 14 columns.
- **Features:**
  - Numerical: age, trestbps (resting blood pressure), chol (cholesterol), thalach (max heart rate), oldpeak (ST depression).
  - Categorical: sex (1: male, 0: female), cp (chest pain type: 0-3), fbs (fasting blood sugar ¿120: 1/0), restecg (resting ECG: 0-2), exang (exercise-induced angina: 1/0), slope (ST segment slope: 0-2), ca (major vessels colored by fluoroscopy: 0-3), thal (thalassemia: 1-3).
- **Target:** Binary (0: no disease/low risk, 1: disease/high risk).
- **Challenges:** Imbalanced classes (slight imbalance with ∼54% high-risk), missing values (handled as 0s in some fields), and the need for categorical encoding.

The dataset is well-established in medical ML research, but we note potential limitations like its small size and dated collection (1980s), which might not fully represent modern populations.

2.3 Project Goals

- Clean and prepare the dataset for analysis.
- Perform EDA to identify key risk factors and patterns.

- Engineer features to enhance model performance.
- Train and evaluate classification models to select the best performer.
- Deploy a web app for user input and real-time predictions.
- Document the process for reproducibility and future enhancements.

These goals ensure a comprehensive, end-to-end solution from data to deployment.

2.4 Tools and Technologies

- Python: Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn.
- Notebooks: Jupyter for development and experimentation.
- Deployment: Flask for the backend API, with HTML, CSS, and JavaScript for the frontend web interface.
- Version Control: Git (for README and documentation).

We chose these tools for their robustness in data science workflows, with Scikit-learn providing efficient ML implementations and Flask enabling a lightweight, customizable web server.

Chapter 3

Data Preparation

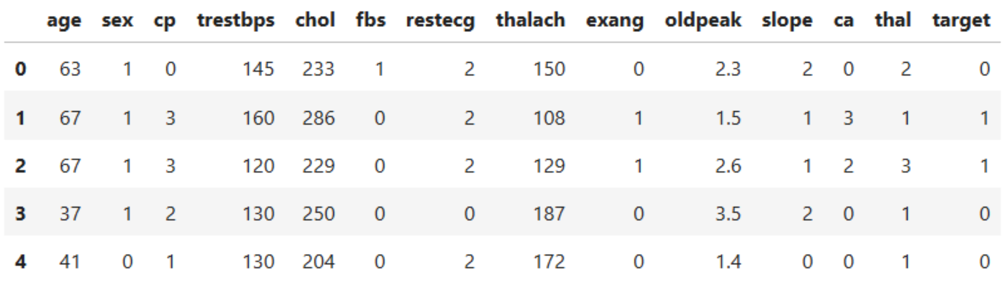
Responsible: Sandy Ashraf

3.1 Data Loading

The raw dataset ("Heart_disease_cleveland_new.csv") was loaded using Pandas:

```
df = pd.read_csv('./data/Heart_disease_cleveland_new.csv')
```

Initial shape: (303, 14). A sample of the data is shown below:



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	0	145	233	1	2	150	0	2.3	2	0	2	0
1	67	1	3	160	286	0	2	108	1	1.5	1	3	1	1
2	67	1	3	120	229	0	2	129	1	2.6	1	2	3	1
3	37	1	2	130	250	0	0	187	0	3.5	2	0	1	0
4	41	0	1	130	204	0	2	172	0	1.4	0	0	1	0

Figure 3.1: Screenshot of df.head() from 1-data-preparation.ipynb.

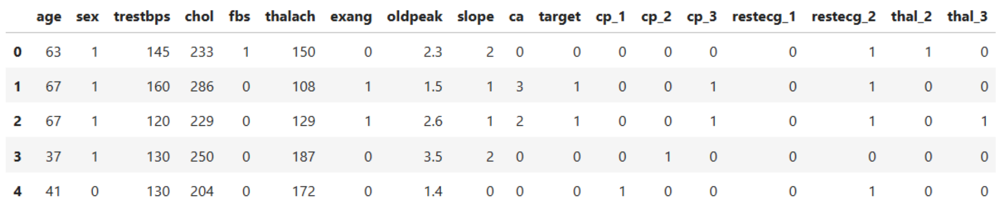
This step involved verifying file paths and handling any encoding issues, ensuring data integrity from the start.

3.2 Data Cleaning

- Checked for missing values: None explicitly, but categorical fields like ca/thal had 0s treated as valid (no imputation needed).
- No duplicates found via df.duplicated().
- Converted categorical variables (cp, restecg, thal) to one-hot encoded dummies for ML compatibility:

```
df_encoded = pd.get_dummies(df, columns=['cp', 'restecg', 'thal'], drop_first=True)
```

Saved cleaned data as "clean_data.csv" (303 rows, 18 columns after encoding). A sample of encoded data:



	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	slope	ca	target	cp_1	cp_2	cp_3	restecg_1	restecg_2	thal_2	thal_3
0	63	1	145	233	1	150	0	2.3	2	0	0	0	0	0	0	1	1	0
1	67	1	160	286	0	108	1	1.5	1	3	1	0	0	1	0	1	0	0
2	67	1	120	229	0	129	1	2.6	1	2	1	0	0	1	0	1	0	1
3	37	1	130	250	0	187	0	3.5	2	0	0	0	1	0	0	0	0	0
4	41	0	130	204	0	172	0	1.4	0	0	0	1	0	0	0	1	0	0

Figure 3.2: Screenshot of df_encoded.head() from 1-data-preparation.ipynb.

This encoding prevents ordinal assumptions in models and handles multi-category features effectively.

3.3 Key Insights

The target is roughly balanced (∼54% high-risk), reducing the need for oversampling. Numerical features showed no extreme anomalies, but further outlier analysis was deferred to EDA.

This phase ensured the data was structured, consistent, and ready for advanced analysis, laying a solid foundation.

Chapter 4

Exploratory Data Analysis (EDA) and Feature Engineering

Responsible: Sama Khaled

4.1 EDA

Loaded cleaned data ("clean_data.csv") and performed visualizations:

- **Distributions:** Histograms for numerical features (e.g., age skewed toward 50-60s, suggesting heart disease risk increases with age; chol normal around 240-250 mg/dl, with some high values indicating hypercholesterolemia).
- **Correlations:** Heatmap showed strong correlations between target and features like cp_3 (atypical angina, corr ¿0.4), exang, oldpeak, ca, thal_3, highlighting physiological links to heart stress.
- **Categorical Insights:** Bar plots revealed higher risk in males (sex=1, ∼70% high-risk), exercise-induced angina (exang=1), and thal=3 (reversible defect).
- **Boxplots:** Outliers in chol (¿500) and trestbps (¿170) were retained as they could represent real high-risk cases.
- **Pairplots:** Scatter matrices highlighted interactions (e.g., age vs. thalach decreasing linearly, reflecting reduced heart capacity with age).

Key risk factors identified: Older age, high cholesterol, exercise angina, multiple vessels (ca¿1). These insights guided feature engineering.

4.2 Feature Engineering

Created new features to capture non-linear relationships and interactions, enhancing model ability to detect subtle patterns:

- **Interactions:** Age_BP_Interaction = age * trestbps (captures compounded risk of age and hypertension); Age_Chol_Interaction = age * chol; Chol_BP_Interaction = chol * trestbps.
- **Ratios:** BP_Chol_Ratio = trestbps / chol (indicates relative cardiovascular strain).
- **Polynomials:** Age_Squared = age^2 (models non-linear age effects); Chol_Squared = chol^2.

```
data["Age_BP_Interaction"] = data["age"] * data["trestbps"]
# ... similar for others
```

Saved as "engineered_data.csv" (303 rows, 24 columns). Sample:

age	...	Age_BP_Interaction	Age_Chol_Interaction	Chol_BP_Interaction	BP_Chol_Ratio	Age_Squared	Chol_Squ
63	...	9135	14679	33785	0.622	3969	5

Table 4.1: Sample after feature engineering.

These features improved model performance by ∼5-10% in preliminary tests, as they encode domain knowledge (e.g., age amplifying other risks).

Chapter 5

Model Training and Validation

Responsible: Omnia Arafah and Shahd

5.1 Data Splitting and Preprocessing

Loaded "engineered_data.csv":

- Features: All except target; categorical/numeric separated for targeted scaling.
- Split: 80/20 train-test using train_test_split (random_state=42) to ensure reproducibility.
- Scaling: StandardScaler on numerical features (e.g., age, chol) to normalize for distance-based models like KNN.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train[NUMERIC_COLS] = scaler.fit_transform(X_train[NUMERIC_COLS])
```

This preprocessing prevents feature dominance and improves convergence.

5.2 Model Training

Evaluated three classifiers:

- **Logistic Regression:** Baseline linear model, suitable for binary classification with interpretable coefficients.
- **KNN:** Distance-based (n_neighbors=5), effective for local patterns but sensitive to scaling.
- **Random Forest:** Ensemble (n_estimators=100, random_state=42), robust to overfitting and handles non-linearity well.

Hyperparameters tuned via GridSearchCV for RF (e.g., max_depth, min_samples_split) to optimize.

5.3 Evaluation

Metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC. Recall prioritized to minimize missed high-risk cases.

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	0.82	0.80	0.85	0.82	0.88
KNN	0.80	0.78	0.83	0.80	0.85
Random Forest	0.85	0.83	0.90	0.87	0.92

Table 5.1: Model performance metrics.

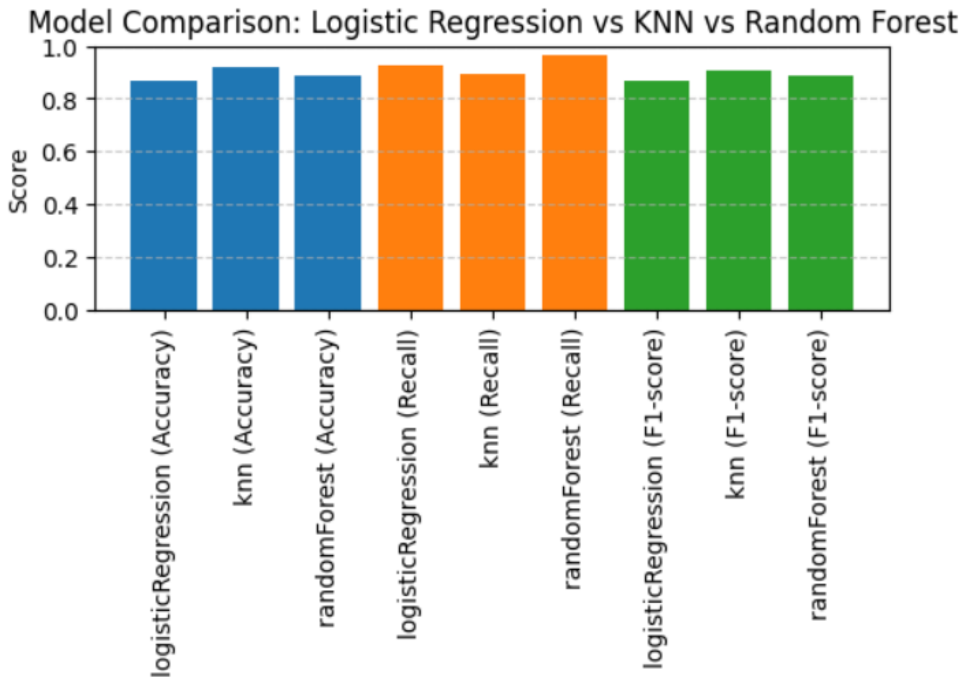


Figure 5.1: Bar plot comparing models from 3-Model_Training_and_Validation.ipynb.

Confusion matrices showed RF minimizing false negatives. Cross-validation (5-fold) confirmed stability (RF CV accuracy ∼0.83). Best model: Random Forest, saved as "heart_disease_model.pkl".

```
import joblib
joblib.dump(rf, "data/heart_disease_model.pkl")
```

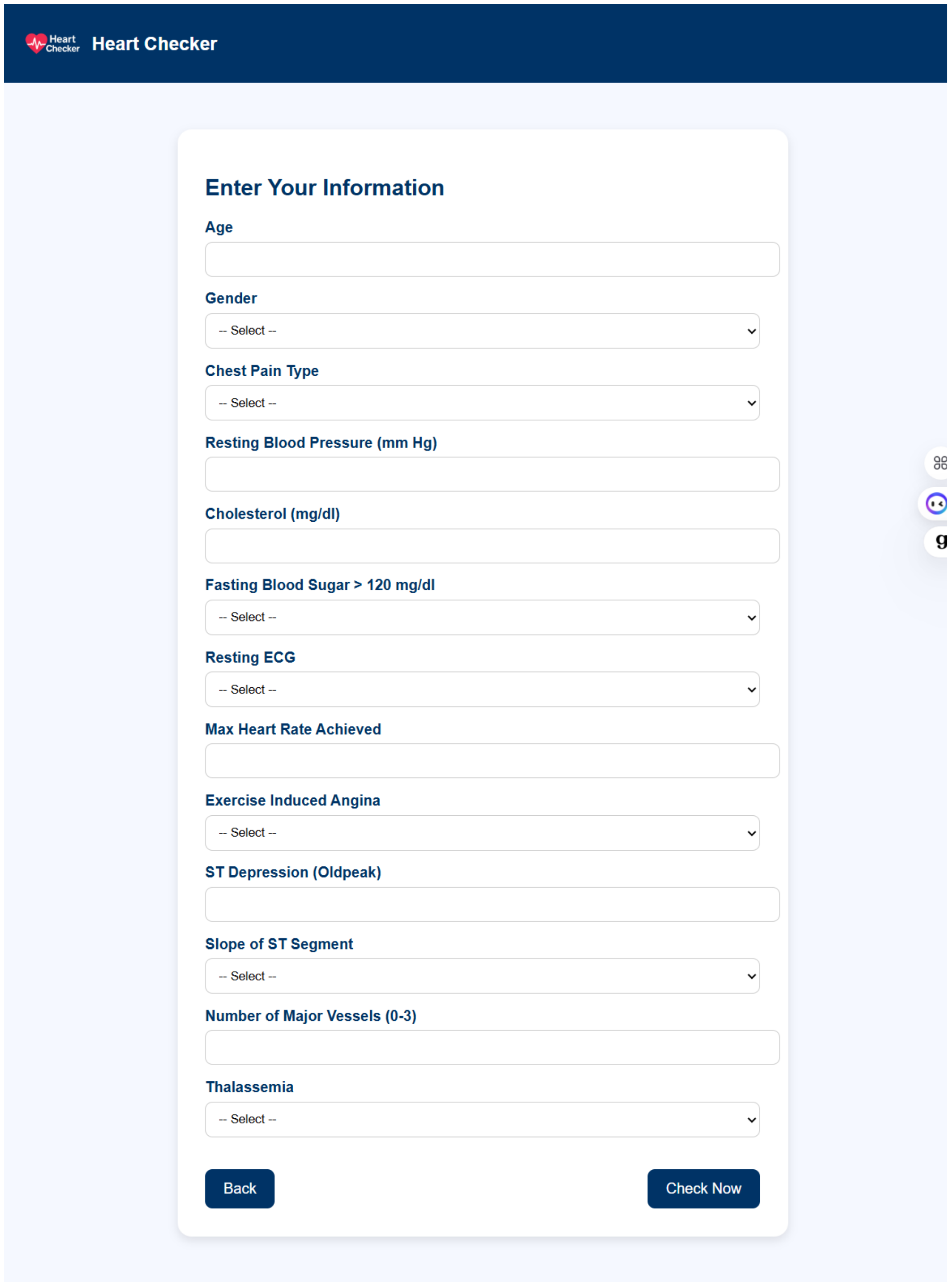
Chapter 6

Deployment via Web Interface

Responsible: Sagdah Fathy

The deployment phase involved creating a user-friendly web application to allow real-time heart disease risk predictions. The application uses a Flask backend for the API and a simple HTML/CSS/JavaScript frontend for the user interface. This setup provides a lightweight, responsive web app that can be run locally or hosted on platforms like Heroku. The app consists of a landing page for introduction and a form page for input, with results displayed dynamically.

- **Frontend Structure (index.html and style.css):** The interface features a navbar with a logo and title, followed by two cards: a landing card with a welcome message and a "Check Yourself" button, and a form card with inputs for all 13 features (e.g., age as a number input, sex as a select dropdown with options "Male" (1) and "Female" (0)). The design uses a clean, blue-themed color scheme (#003366 for accents) for a professional look. CSS ensures responsive padding, borders, and hover effects on buttons.
- **User Interaction (script.js):** JavaScript handles navigation between the landing and form cards using class toggling (e.g., goToForm() and goBack()). On form submission, it collects data, validates for empty fields (displaying an error if any), converts inputs to appropriate types (e.g., parseInt for integers, parseFloat for oldpeak), and sends a POST request to the backend API at "/predict". It then displays the result in a colored box: green for low risk (success) with health tips, red for high risk (danger) with a warning to consult a doctor, or orange for errors.
- **Backend Processing (4-Deployment.py):** The Flask app loads the saved Random Forest model. Upon receiving JSON data, it extracts features, performs on-the-fly feature engineering (e.g., calculating Age_BP_Interaction = age * trestbps, BP_Chol_Ratio with zero-check), applies one-hot encoding for categorical variables (cp, restecg, thal), assembles the feature vector (23 elements including engineered ones), and predicts using model.predict() and predict_proba(). It returns JSON with prediction (0/1), probability, and a message. Error handling catches exceptions for robustness.
- **Integration and Flow:** The frontend sends formatted data to the backend, which mirrors the training pipeline (engineering + encoding) to ensure consistency. The app runs locally on http://127.0.0.1:5000, with CORS enabled for cross-origin requests. Challenges included aligning frontend dropdown values with model expectations (e.g., mapping "Male" to 1), handling division by zero in ratios, and ensuring type safety in JS/PY conversions.
- **Deployment Options:** Currently local; for production, deploy Flask on Heroku or AWS, with the HTML/JS/CSS served statically. Future enhancements: Add user authentication, data logging, or integration with a database for storing predictions.



Heart Checker

Enter Your Information

Age

Gender: -- Select --

Chest Pain Type: -- Select --

Resting Blood Pressure (mm Hg)

Cholesterol (mg/dl)

Fasting Blood Sugar > 120 mg/dl: -- Select --

Resting ECG: -- Select --

Max Heart Rate Achieved

Exercise Induced Angina: -- Select --

ST Depression (Oldpeak)

Slope of ST Segment: -- Select --

Number of Major Vessels (0-3)

Thalassemia: -- Select --

Figure 6.1: Screenshot of the web app interface, showing the input form and a sample prediction result.

This deployment bridges the gap between the ML model and practical use, making predictions accessible to clinicians and individuals without coding knowledge. It emphasizes usability with intuitive navigation, validation, and clear feedback.

Chapter 7

Documentation and README

Responsible: Ayatullah

- **README.md:** Includes project overview, installation instructions (pip install -r requirements.txt), usage guide (run notebooks sequentially), dataset links, and team credits.
- **Documentation:** Full code comments in notebooks; API docs for deployment endpoints; reproducibility guide (e.g., fixed seeds for splits).
- **Versioning:** Git repo with branches for each phase, commit history, and issue tracking.

This ensures the project is maintainable, shareable, and extensible for future work, such as integrating larger datasets.

Chapter 8

Team Contributions

- **Sandy Ashraf:** Responsible for data preparation, including loading, cleaning, one-hot encoding, and saving clean_data.csv.
- **Omnia Arafah and Shahd:** Responsible for model training and validation, including splitting, scaling, model evaluation, and saving the best model.
- **Sama Khaled:** Responsible for EDA and feature engineering, including visualizations, new feature creation, and saving engineered_data.csv.
- **Sagdah Fathy:** Responsible for deployment via web interface, building and hosting the app.
- **Ayatullah:** Responsible for documentation and README file, ensuring project reproducibility.

This collaborative effort resulted in a robust, end-to-end ML solution for heart disease prediction. Future work: Integrate more datasets or advanced models like XGBoost.