

Heart Disease Prediction — End-to-End MLOps Case Study

Date: January 06, 2026

This report summarizes the MLOps implementation for a Heart Disease Prediction system and appends evidence images covering EDA, model evaluation, and API endpoints.

1. Project Overview

This project implements an end-to-end MLOps pipeline for a Heart Disease Prediction system. The objective is to demonstrate best practices across data processing, model training, experiment tracking, CI/CD, containerization, deployment, and monitoring.

2. Setup & Installation Instructions

- OS: Linux VM (Docker + Minikube)
- Python: 3.9
- Tools: Docker, Minikube, kubectl, GitHub Actions

Steps:

1. git clone
2. cd mllops-heart
3. docker build -t heart-api:latest .
4. docker run -p 8000:8000 heart-api:latest
5. minikube start --driver=docker
6. minikube image load heart-api:latest
7. kubectl apply -f k8s/

3. Exploratory Data Analysis (EDA) & Modeling Choices

- Dataset: Heart Disease dataset (tabular clinical features)
- EDA: Missing value checks, feature distributions, correlation analysis
- Model: RandomForestClassifier
- Scaling: StandardScaler
- Threshold: Probability $\geq 0.5 \rightarrow$ Disease classification

Random Forest was chosen for its robustness and interpretability on tabular data.

4. Experiment Tracking Summary

- Tool: MLflow (local tracking)
- Tracked Parameters: model type, number of estimators
- Metrics: accuracy, probability score
- Artifacts: trained model (.pkl), scaler (.pkl)

MLflow enables reproducibility and comparison across experiments.

5. Architecture Diagram (Logical Flow)

User → FastAPI → Preprocessing → ML Model → Prediction

↑ Docker

↑ Kubernetes (Minikube)

↑ CI/CD (GitHub Actions)

6. CI/CD Pipeline Workflow

GitHub Actions Pipeline:

- Linting (flake8)
- Unit tests (pytest)
- Dependency installation
- Model training
- Artifact storage

Each commit triggers automated validation ensuring code quality and reproducibility.

7. Deployment Workflow

- Dockerized FastAPI application
- Kubernetes Deployment + NodePort Service
- Exposed endpoints: /health, /predict

Deployment validated using Minikube service URL.

8. Monitoring & Logging

- Logging integrated using Python logging module
- Logs include request payload, prediction output, and timestamps
- Logs verified via: docker logs and kubectl logs

This demonstrates basic production observability.

9. Code Repository

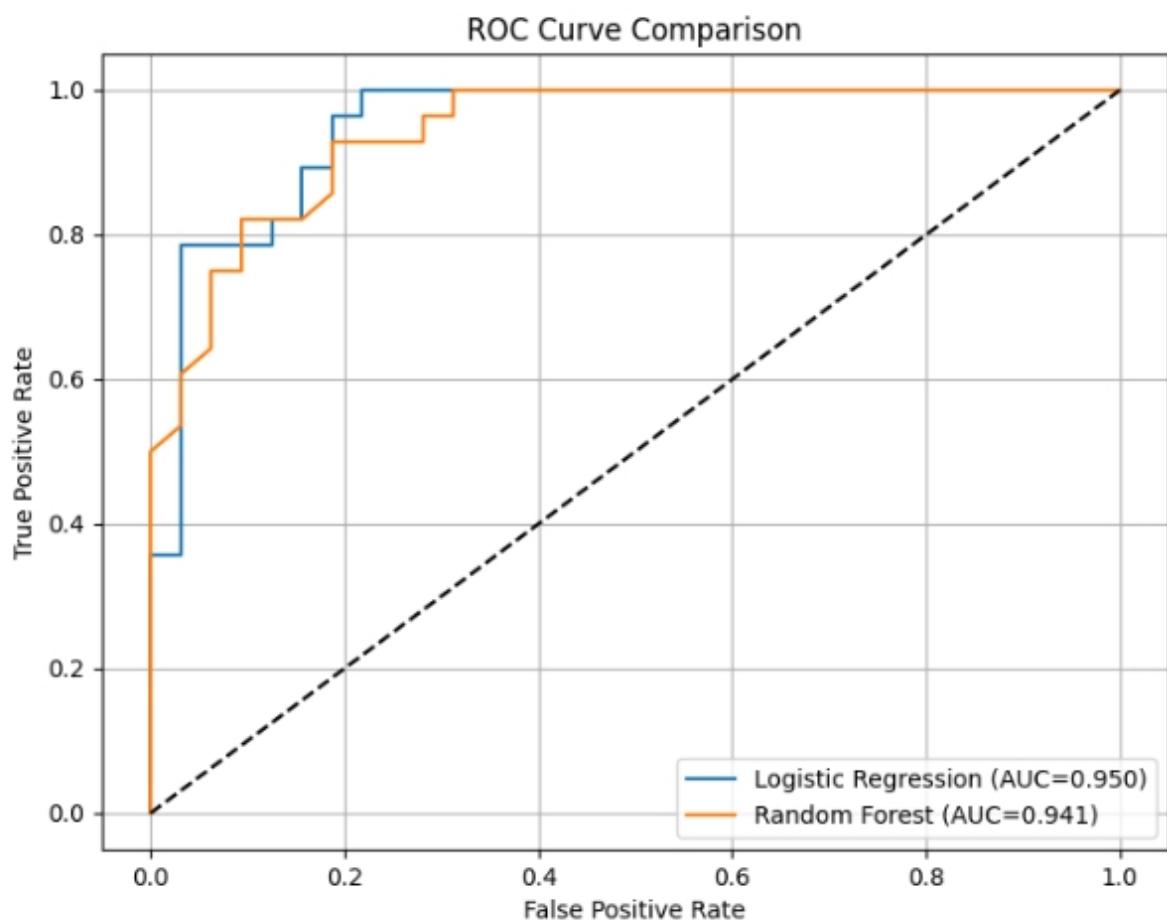
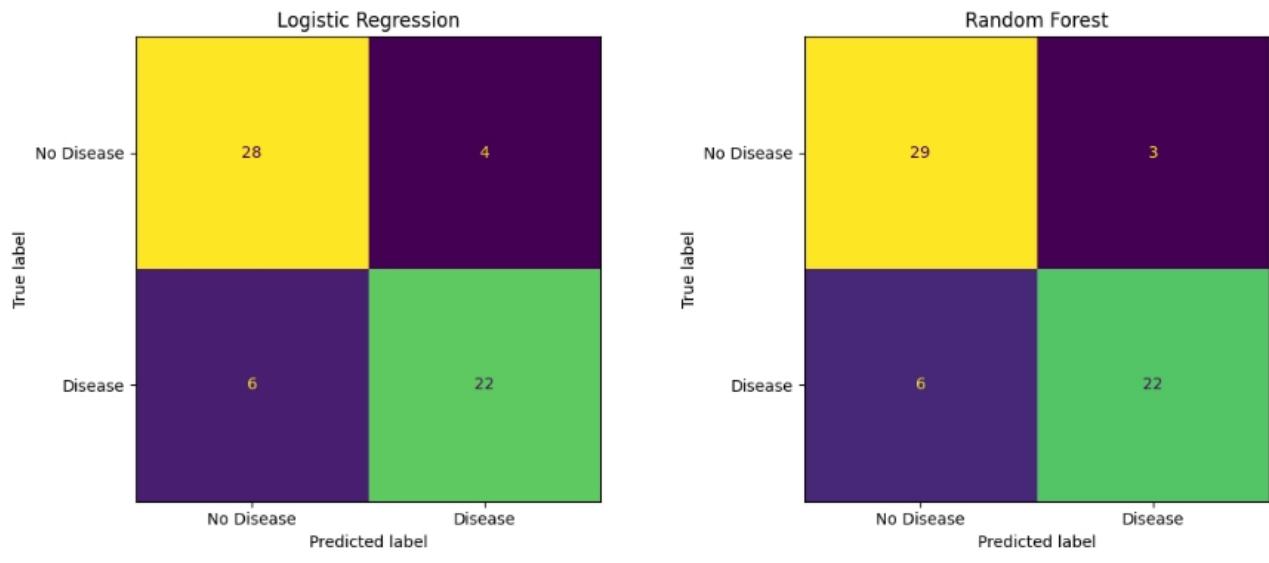
GitHub Repository: <https://github.com/Jigyansh87/mlops-heart>

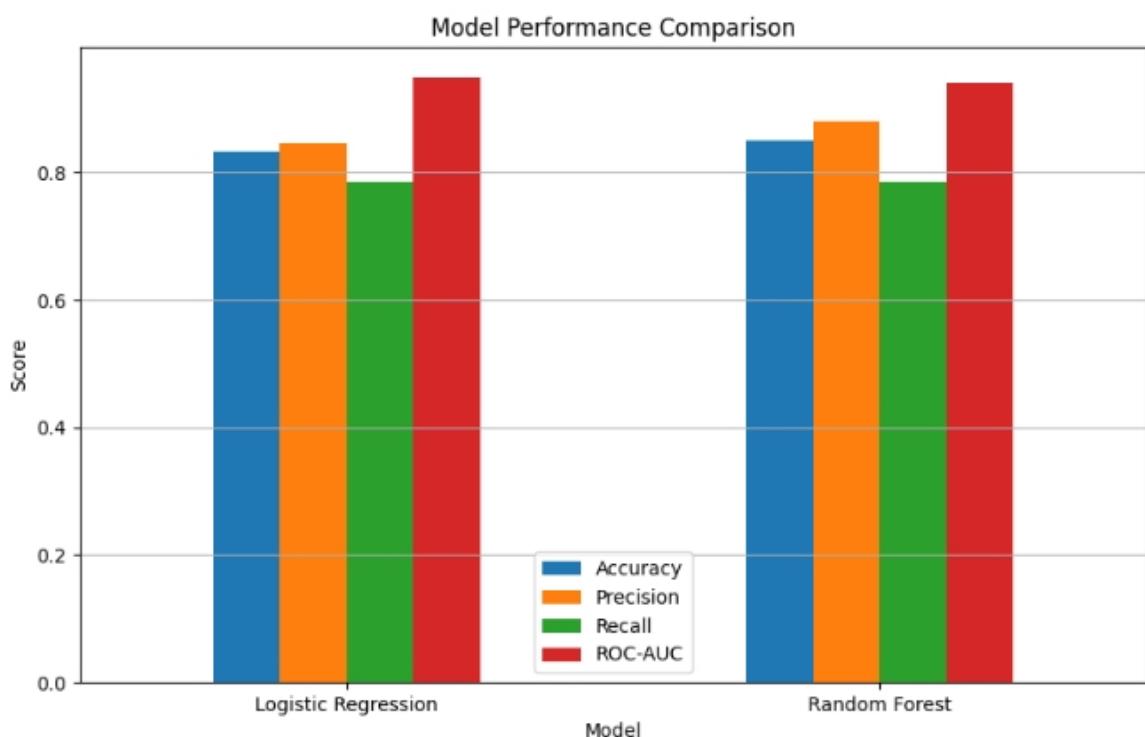
10. Conclusion

This project successfully demonstrates a complete MLOps lifecycle from development to deployment.

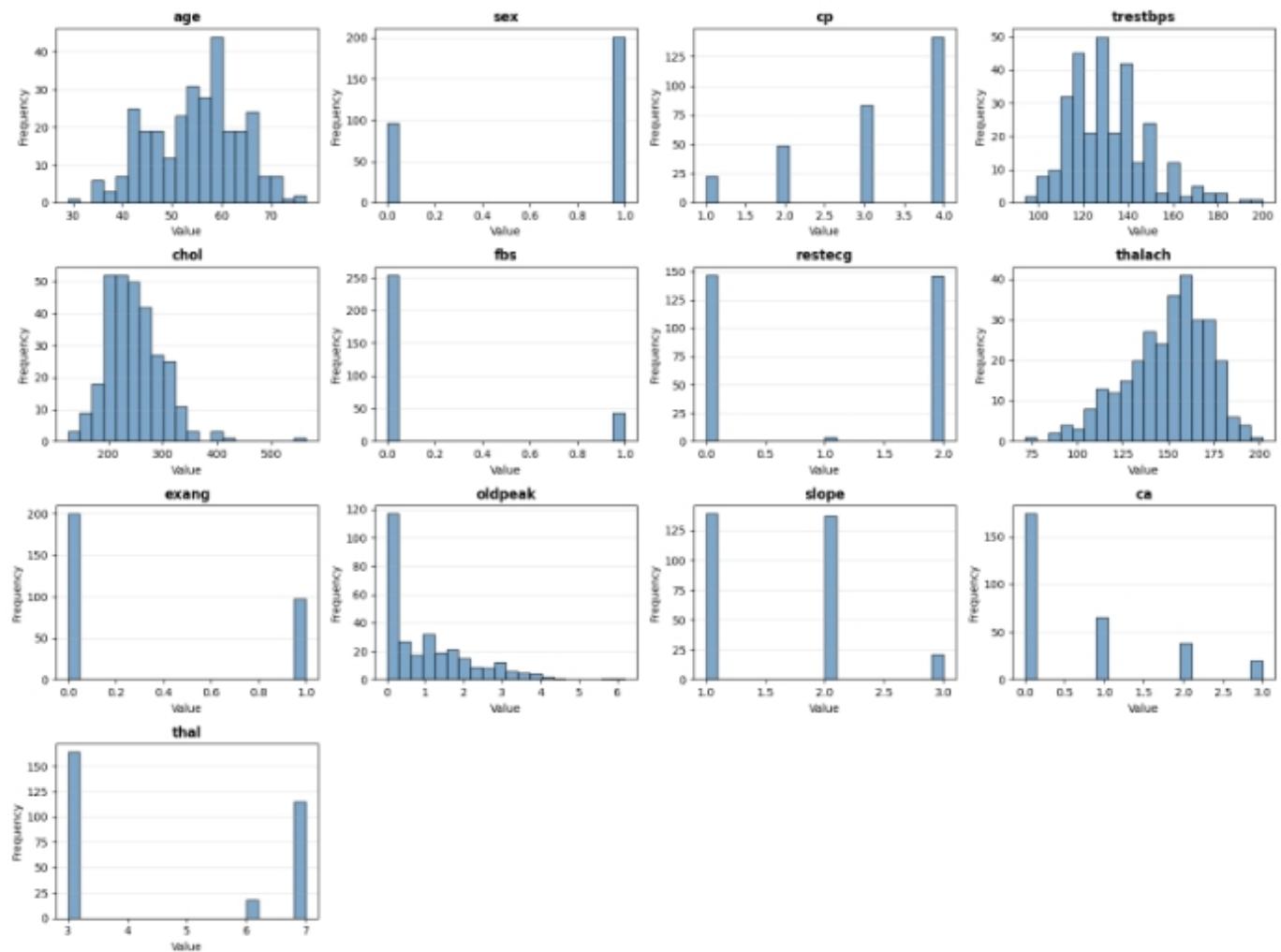
Evidence

The following images provide visual evidence of EDA results, model evaluation, class balance, and API Swagger screenshots.

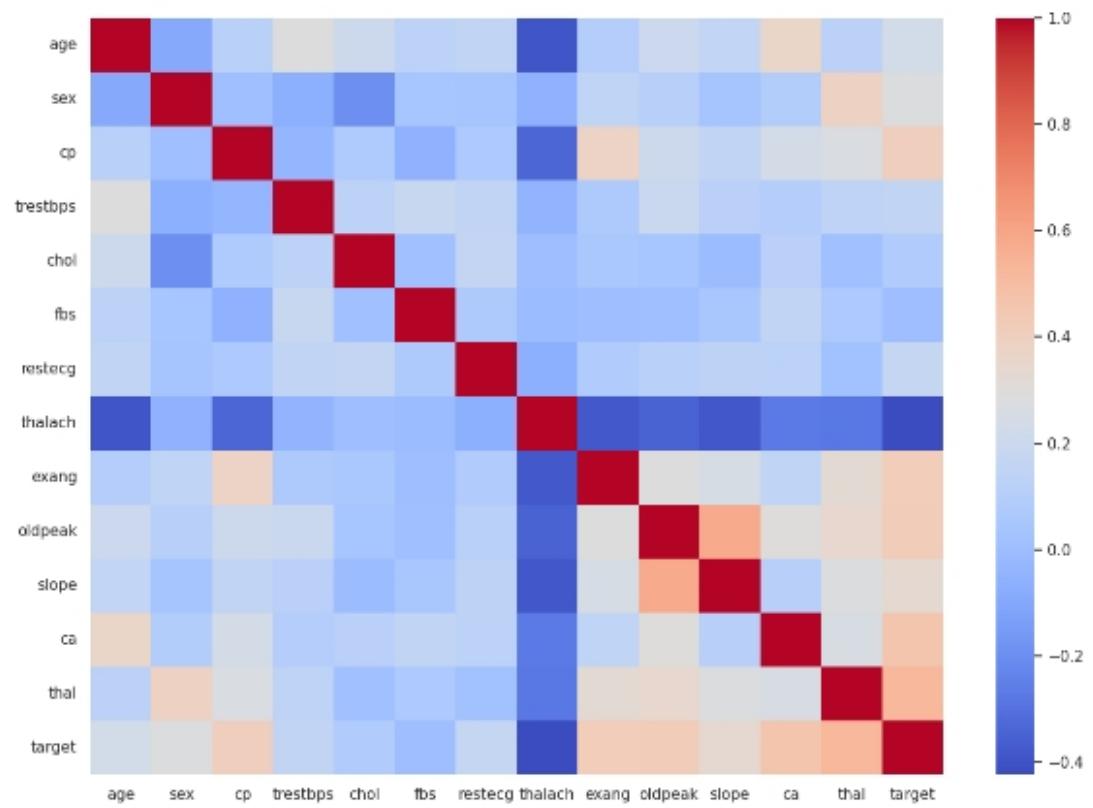




Model Performance Comparison: Accuracy, Precision, Recall, ROC-AUC



Feature Distributions (Histograms) for Clinical Variables



Correlation Heatmap of Features and Target



Class Distribution (Target Balance)

The screenshot shows the Swagger UI interface for a Heart Disease Prediction API. The URL is http://localhost:8000/docs#/heartdisease/predict_post. The top navigation bar has tabs for 'POST' and '/predict'. Below it, there's a 'Parameters' section with 'No parameters' listed. The 'Request body' section is marked as required and set to 'application/json'. It contains a JSON editor with the following content:

```
{ "features": [67.0, 1.0, 4.0, 160.0, 286.0, 0.0, 2.0, 386.0, 1.0, 1.5, 2.0, 3.0, 3.0] }
```

Below the request body, there are 'Execute' and 'Clear' buttons. The 'Responses' section is collapsed. Under 'curl', there is a code snippet:

```
curl -X 'POST' \
  'http://localhost:8000/predict' \
  -H 'Content-Type: application/json' \
  -d '{ "features": [67.0, 1.0, 4.0, 160.0, 286.0, 0.0, 2.0, 386.0, 1.0, 1.5, 2.0, 3.0, 3.0] }'
```

The 'Request URL' field contains <http://localhost:8000/predict>. The 'Server response' section is collapsed.

Swagger UI: /predict endpoint demonstration

The screenshot shows the Swagger UI interface for a metrics endpoint. The URL is http://localhost:8000/docs#/default/metrics_get. The top navigation bar has tabs for 'GET' and '/metrics'. Below it, there's a 'Parameters' section with 'No parameters' listed. The 'Request body' section is collapsed. The 'Responses' section is collapsed. Under 'curl', there is a code snippet:

```
curl -X 'GET' \
  'http://localhost:8000/metrics' \
  -H 'Accept: application/json'
```

The 'Request URL' field contains <http://localhost:8000/metrics>. The 'Server response' section is collapsed. The 'Code' and 'Details' tabs are visible under the responses section. The '200' tab is selected, showing the 'Response body' and 'Response headers' sections. The 'Response body' contains the following JSON:

```
{ "requests_served": 2, "latency_ms": 2656.74 }
```

The 'Response headers' section shows the following headers:

```
content-length: 47
content-type: application/json
date: Tue, 06 Jan 2020 10:14:37 GMT
server: unicorn
```

Swagger UI: /metrics endpoint output

Heart Disease Prediction API

http://localhost:8000/docs/default/health_get

openapi.json

default

Get /health

Parameters

No parameters

Executes Clear

Responses

curl -X "GET" "\n \"http://localhost:8080/health\"\n -H \"Accept: application/json\""

Request URL

http://localhost:8080/health

Server response

Code Details

200 Response body

```
{\n    \"status\": \"ok\"\n}
```

Download

Response headers

```
content-length: 15\ncontent-type: application/json\ndate: Tue, 06 Jan 2020 13:18:42 GMT\nserver: unicorn
```

Responses

Code Description

200 Successful Response

Metadata

Links

The screenshot shows the Swagger UI interface for a 'Heart Disease Prediction API'. The top navigation bar has tabs for 'Home', 'API', 'Models', and 'Metrics'. The current view is under the 'API' tab, specifically for the '/health' endpoint. The endpoint is defined as a GET method. There are no parameters listed. The 'Responses' section shows a successful 200 OK response. The 'curl' command provided is used to fetch the health endpoint from the local host. The 'Request URL' is also shown. The 'Server response' section contains the JSON response body and its corresponding HTTP headers. Below this, the 'Responses' table lists the 200 OK response again with a 'Successful Response' description and a 'Metadata' link.

Swagger UI: /health endpoint output