

# MLOps Assignment Report: Heart Disease Prediction

## Contributors

GROUP 33

- ARYAMANN SINGH - 2024aa05025
- ANANTHAN P - 2024aa05692
- BALAJI R - 2024aa05844
- BALSURE ANIKET K - 2024aa05296
- SAURAV BANSAL - 2023aa05710

## Demo Video

[YouTube URL](#)

## Repository Link (with setup)

[GitHub Repository](#)

## 1. Introduction

This report details the implementation of an end-to-end MLOps pipeline for heart disease risk prediction using the UCI Heart Disease dataset. The project demonstrates modern MLOps practices including data processing, model development, experiment tracking, CI/CD, containerization, and deployment.

## 2. Dataset and Data Acquisition

**Dataset:** UCI Heart Disease Dataset

- **Source:** UCI Machine Learning Repository
- **Features:** 13 clinical features + target

- **Samples:** ~300 instances
- **Target:** Binary (0: no disease, 1: disease)

#### **Data Acquisition:**

- Script: `src/data_prep.py` downloads from UCI URL
- Cleaning: Handle missing values (marked as '?'), convert to binary target
- Storage: `data/raw/heart.csv`

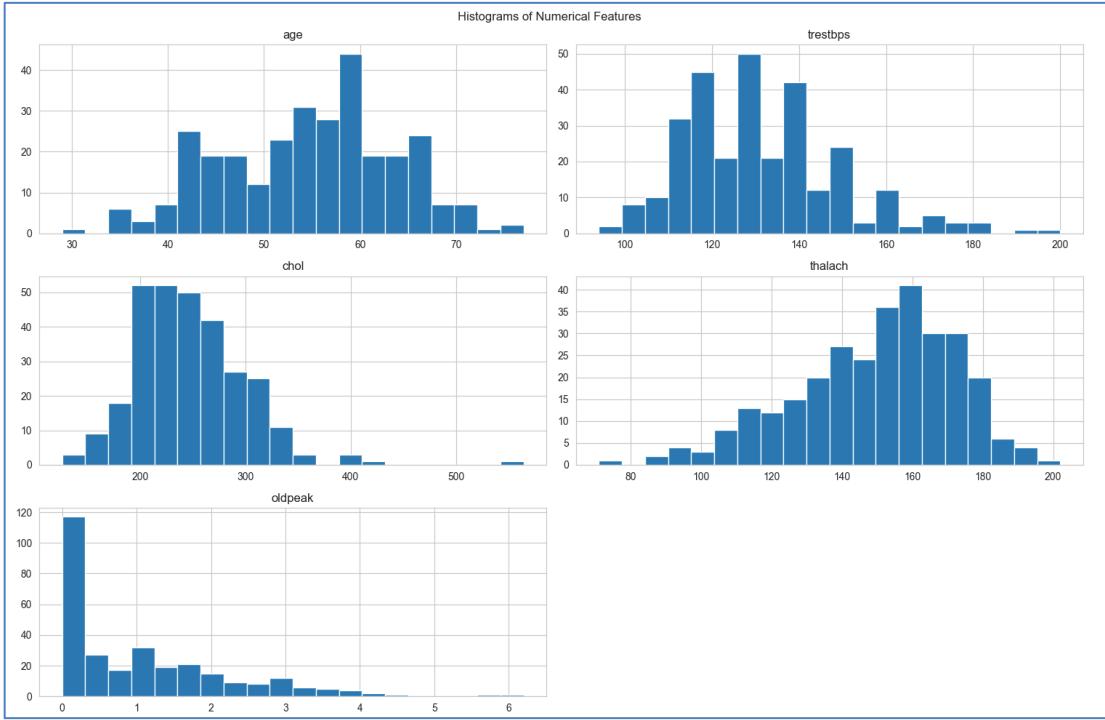
### **3. Exploratory Data Analysis (EDA)**

#### **Key Findings** (see `notebooks/eda.ipynb`):

- **Class Balance:** Slight imbalance (~46% positive cases; 137/297).
- **Correlations:** Strongest correlations with target include **thal** and **ca** (positive) and **thalach** (negative); **cp** and **oldpeak** are also meaningfully associated.
- **Distributions:** Age is roughly bell-shaped; cholesterol shows clear outliers.
- **Categorical Analysis:** Higher chest pain types associated with disease.

#### **Visualizations:**

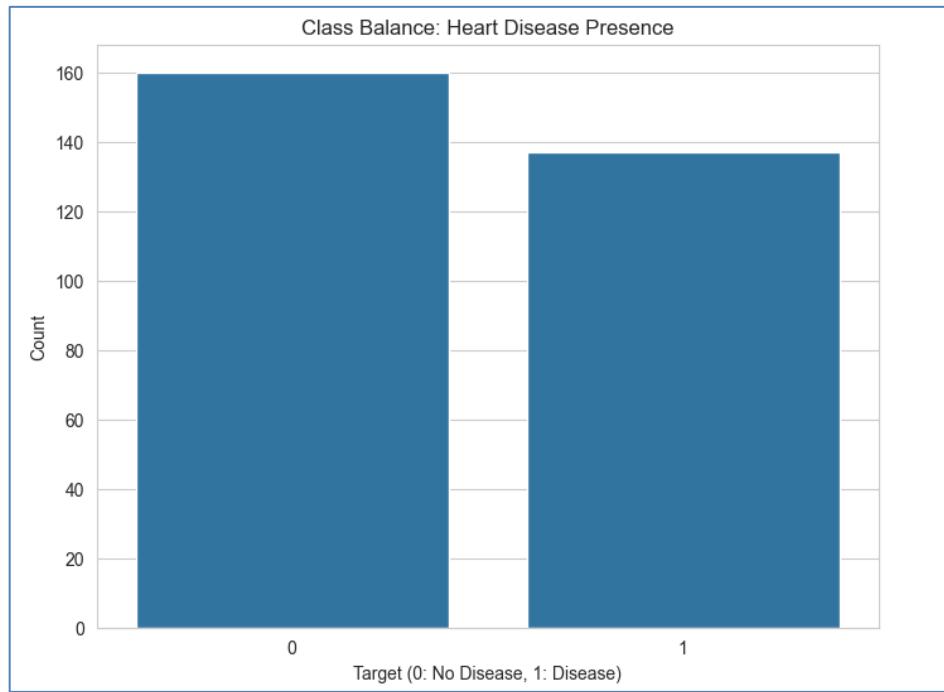
- Histograms for numerical features



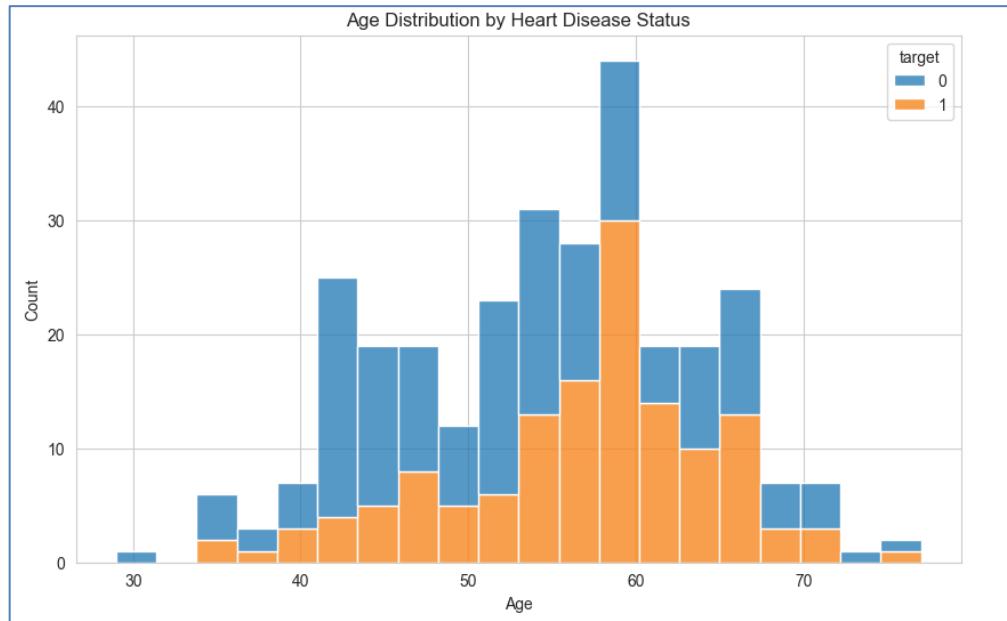
- Correlation heatmap



- Class balance bar plot



- Age distribution by target



## 4. Feature Engineering and Model Development

### Preprocessing Pipeline:

- Numerical: StandardScaler

- Categorical: OneHotEncoder (handle\_unknown='ignore')
- Pipeline: ColumnTransformer + Model

### **Models Evaluated:**

1. Logistic Regression (tuned C, penalty)
2. Random Forest (tuned n\_estimators, max\_depth)

**Hyperparameter Tuning:** GridSearchCV with 5-fold CV, ROC-AUC scoring

### **Evaluation Metrics:**

- ROC-AUC (primary)
- Accuracy, Precision, Recall

**Best Model:** Logistic Regression (ROC-AUC: 0.9113)

## **5. Experiment Tracking**

**Tool:** MLflow

- **Experiments:** Logged parameters, metrics, model artifacts
- **Runs:** Separate runs for each model with best params
- **UI:** `mlflow ui` for visualization

**MLflow experiment runs and metrics**

- MLflow UI - experiment list

heart-disease-experiments Machine learning						
Model attributes			Model attributes		No dataset	Parameters
Model name	Status	Created ↓	Logged from	Source run	roc_auc	model
● model	○ Ready	26 seconds ago	train.py	train_log_reg	0.9112599206349208	log_reg
● model	○ Ready	41 seconds ago	train.py	train_rf	0.8980902777777777	rf
● model	○ Ready	50 seconds ago	train.py	train_rf	0.8980902777777777	rf
● model	○ Ready	1 minute ago	train.py	train_log_reg	0.9112599206349208	log_reg
● model	○ Ready	18 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
● model	○ Ready	19 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
● model	○ Ready	19 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
● model	○ Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg
● model	○ Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg
● model	○ Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg

- MLflow UI - best run details

The screenshot shows the MLflow UI interface for a 'model' named 'Not registered'. The 'Overview' tab is selected. The 'Description' section contains a note: 'No description'. The 'Metrics (1)' section lists a single metric: 'roc\_auc' with a value of '0.9112599206349...'. The 'Parameters (1)' section lists a single parameter: 'model' with a value of 'log\_reg'.

Metric	Dataset	Source run	Value
roc_auc	-	train_log_reg	0.9112599206349...

Parameter	Value
model	log_reg

## 6. Model Packaging and Reproducibility

**Format:** Joblib pickle with sklearn Pipeline

**Dependencies:** requirements.txt with pinned versions

**Reproducibility:** Pipeline ensures consistent preprocessing

## 7. CI/CD Pipeline and Testing

**Tool:** GitHub Actions

**Jobs:**

- Ubuntu: Lint (flake8), test (pytest), data prep, train, upload artifact
- Windows: Test only

**Tests:**

- Data loading: tests/test\_data.py
- Data prep: tests/test\_prep.py

**Artifacts:** Trained model uploaded per run

## 8. Model Containerization

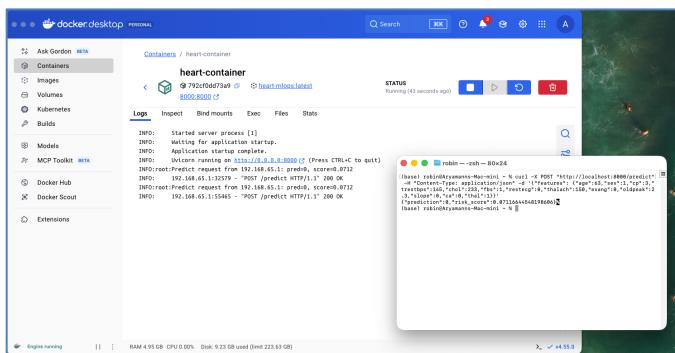
**Tool:** Docker

**Image:** Python 3.10 slim + dependencies

**API:** FastAPI with /predict endpoint

**Testing:** Local build/run with sample input

**Docker build and local container test**



## 9. Production Deployment

**Platform:** Railway (public cloud)

**URL:** <https://heart-mlops-production.up.railway.app>

**Manifests:** Docker-based deployment

**Service:** Web service with automatic scaling

**Verification:** Endpoint testing with curl, deployed API functional

### Railway deployment

- Railway service dashboard

The screenshot shows the Railway service dashboard. On the left, there is a card for the 'heart-mlops' service, which is currently 'Online'. On the right, there is a sidebar titled 'Activity' showing deployment logs:

- heart-mlops - Deployment successful 58 minutes ago
- heart-mlops - Deployment successful 1 hour ago

Below the activity log, there is a message indicating '1 change in heart-mlops' made by 'theveriton' 1 hour ago.

- Deployed /predict response

The screenshot shows a terminal window with the following command and output:

```
(base) robin@Aryamanns-Mac-mini ~ % curl -X POST "https://heart-mlops-production.up.railway.app/predict" -H "Content-Type: application/json" -d '{"features": {"age": 63, "sex": 1, "cp": 3, "trestbps": 145, "chol": 233, "fbs": 1, "restecg": 0, "thalach": 150, "exang": 0, "oldpeak": 2.3, "slope": 0, "ca": 0, "thal": 1}}'  
{"prediction":0,"risk_score":0.07116644548198606}  
(base) robin@Aryamanns-Mac-mini ~ %
```

## 10. Monitoring and Logging

**Logging:** Request logging with client IP, prediction, score

**Metrics:** /metrics endpoint exposes Prometheus-formatted metrics (including predict\_requests\_total)

### Monitoring Stack (Local):

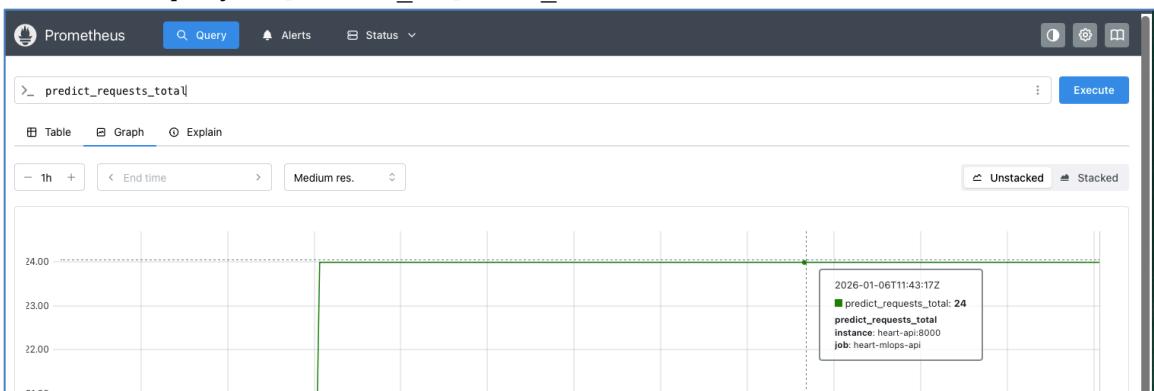
- Prometheus scrapes the API metrics endpoint
- Grafana visualizes metrics from Prometheus

### Monitoring Screenshots

- Prometheus targets UP

The screenshot shows the Prometheus interface under the 'Status' tab, specifically the 'Target health' section. It displays a table with one row for the target 'heart-mlops-api'. The table columns are 'Endpoint', 'Labels', 'Last scrape', and 'State'. The 'Endpoint' column shows 'http://heart-api:8000/metrics'. The 'Labels' column shows 'instance="heart-api:8000"' and 'job="heart-mlops-api"'. The 'Last scrape' column shows '3.12s ago' and '3ms'. The 'State' column shows 'UP' with a green status indicator. There are three input fields at the top: 'Select scrape pool', 'Filter by target health', and 'Filter by endpoint or labels'.

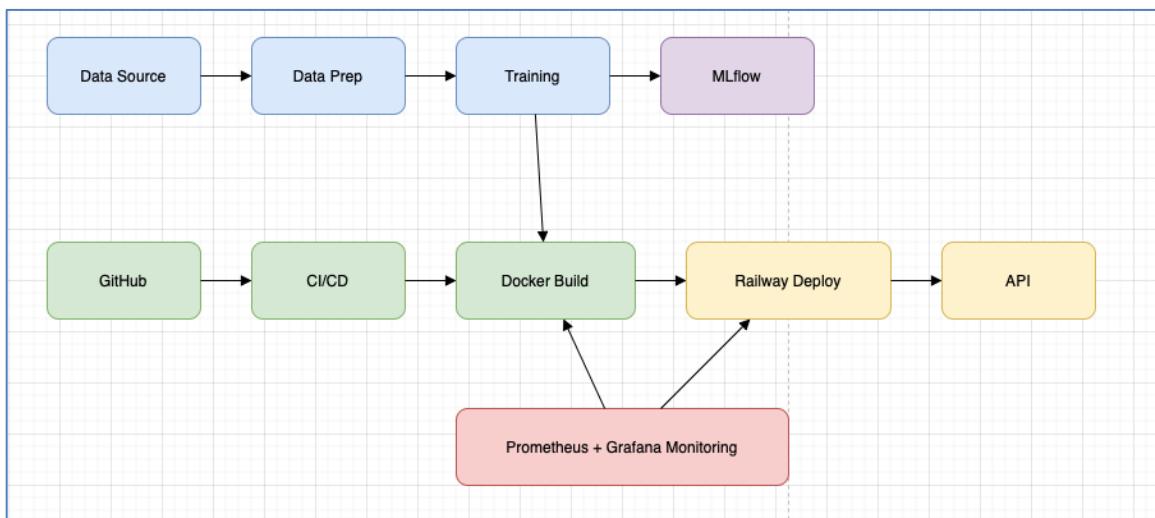
- Prometheus query for predict\_requests\_total



- Grafana dashboard panel showing `predict_requests_total`



## 11. Architecture Diagram



## 12. CI/CD Workflow Screenshots

- Build success:

Triggered via push 1 hour ago

**theveriton pushed → a63753c main**

Status: **Success** | Total duration: **1m 47s** | Artifacts: **1**

**ci.yml**  
on: push

**test-and-train** 59s  
**build-windows** 1m 41s

- Test results:

Run tests

```
1 ▶ Run pytest -q
12 ..
13 2 passed in 0.84s [100%]
```

- Deployment:

**Deployments**

All deployments

Environments

**triumphant-balance / production**

Manage environments

**triumphant-balance / production deployments**

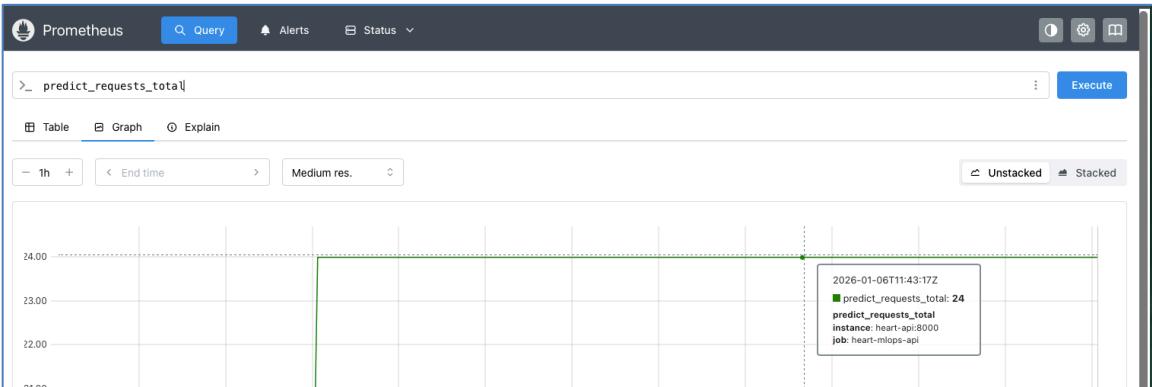
Latest deployments

**triumphant-balance / production**

Last deployed 2 hours ago

<https://railway.com/project/5b308d1e-803e-46b7-bce4-88696b263c2a?environmentId=4e5a7fef-bba8-47b8-99a1-3ceac946bb3a>

- Monitoring:





- MLflow:

The figure is a screenshot of the MLflow UI. At the top, it says "mlflow 3.8.1". Below that is a breadcrumb trail: Home > Dashboards > Number of requests. The main content area is titled "heart-disease-experiments" and "Machine learning". There are tabs for "Runs", "Models" (which is selected and highlighted in blue), and "Traces". Below the tabs is a table with the following data:

Model name	Status	Created	Logged from	Source run	No dataset	Parameters
model	Ready	26 seconds ago	train.py	train_log_reg	0.9112599206349208	log_reg
model	Ready	41 seconds ago	train.py	train_rf	0.8980902777777777	rf
model	Ready	50 seconds ago	train.py	train_rf	0.8980902777777777	rf
model	Ready	1 minute ago	train.py	train_log_reg	0.9112599206349208	log_reg
model	Ready	18 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
model	Ready	19 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
model	Ready	19 hours ago	train.py	train_log_reg	0.9112599206349208	log_reg
model	Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg
model	Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg
model	Ready	19 hours ago	train.py	train_log_reg	0.9025876322751323	log_reg

## 13. Conclusion

The project successfully implements all MLOps requirements with automated pipelines, reproducible models, and production-ready deployment on Railway. Key achievements include hyperparameter tuning, experiment tracking, containerization, and public cloud deployment.