

MLOps Assignment Report: Heart Disease Prediction

Contributors

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

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 (54% positive cases)
- **Correlations:** Strong correlations between thalach (heart rate) and target, cp (chest pain) and target
- **Distributions:** Age follows normal distribution, cholesterol shows 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)

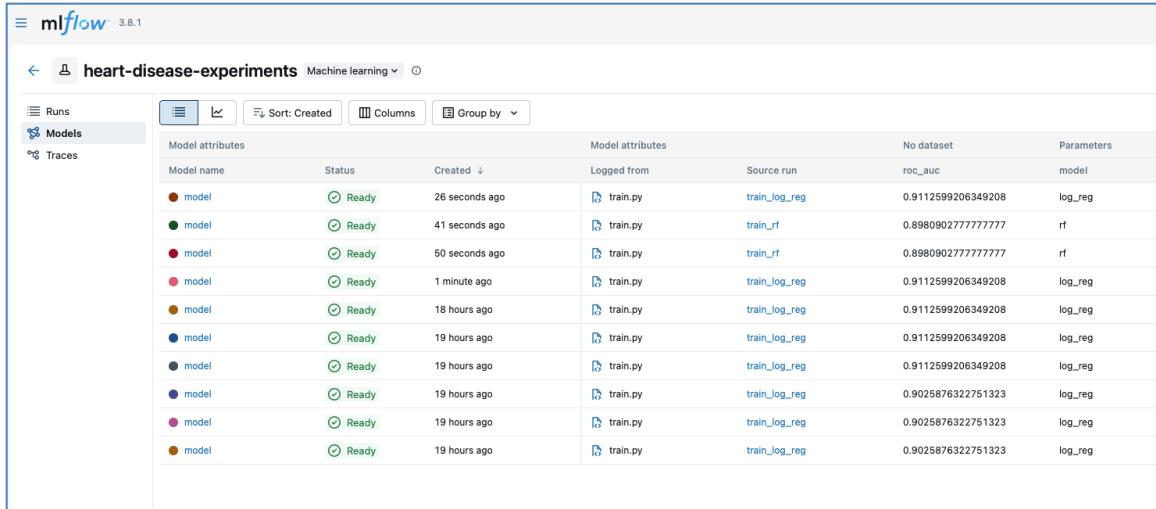
5. Experiment Tracking

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



The screenshot shows the MLflow UI interface with the title "heart-disease-experiments" and "Machine learning". The left sidebar has tabs for "Runs", "Models" (which is selected), and "Traces". The main area displays a table of experiment runs. The columns are: Model name, Status, Created (sorted by created time), Model attributes (Logged from, Source run, No dataset, Parameters). There are 10 rows, each representing a different experiment run. All runs are labeled as "Ready". The "Model attributes" column shows "train.py" for all runs. The "Source run" column shows various names like "train_log_reg", "train_rf", etc. The "No dataset" column shows "roc_auc" and the "Parameters" column shows "model".

Model name	Status	Created ↓	Model attributes	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

- 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

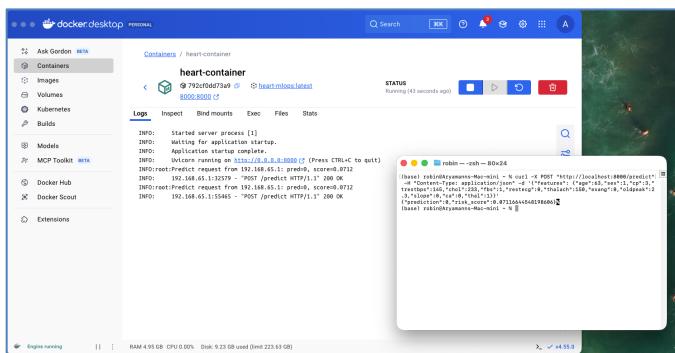
Tool: Docker

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 'Activity' sidebar showing two deployment logs: one successful deployment 58 minutes ago and another successful deployment 1 hour ago. A notification at the bottom indicates '1 change in heart-mlops' by 'theveriton' 1 hour ago.

- Deployed /predict response

The screenshot shows a terminal window titled 'robin -- zsh -- 80x24'. The user runs a curl command to make a POST request to the '/predict' endpoint of the heart-mlops service. The response is a JSON object containing a prediction score of 0.07116644548198606.

```
(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):

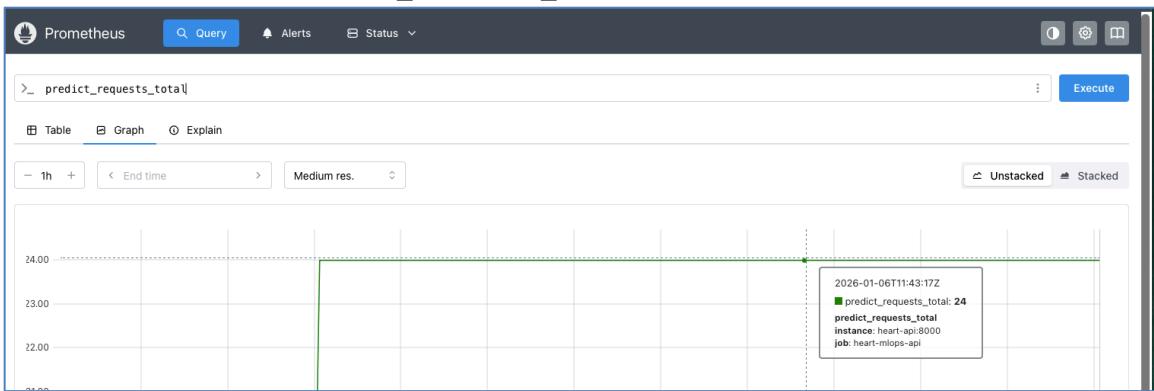
- Prometheus scrapes the API metrics endpoint
- Grafana visualizes metrics from Prometheus

Monitoring Screenshots

- Prometheus targets UP

The screenshot shows the Prometheus Status interface. In the top navigation bar, the 'Status > Target health' tab is selected. Below it, there are three search/filter input fields: 'Select scrape pool', 'Filter by target health', and 'Filter by endpoint or labels'. A table lists a single target: 'heart-mlops-api'. The table columns are 'Endpoint', 'Labels', 'Last scrape', and 'State'. The endpoint is 'http://heart-api:8000/metrics', labels include 'instance="heart-api:8000"' and 'job="heart-mlops-api"', last scrape was 3.12s ago, and the state is 'UP'.

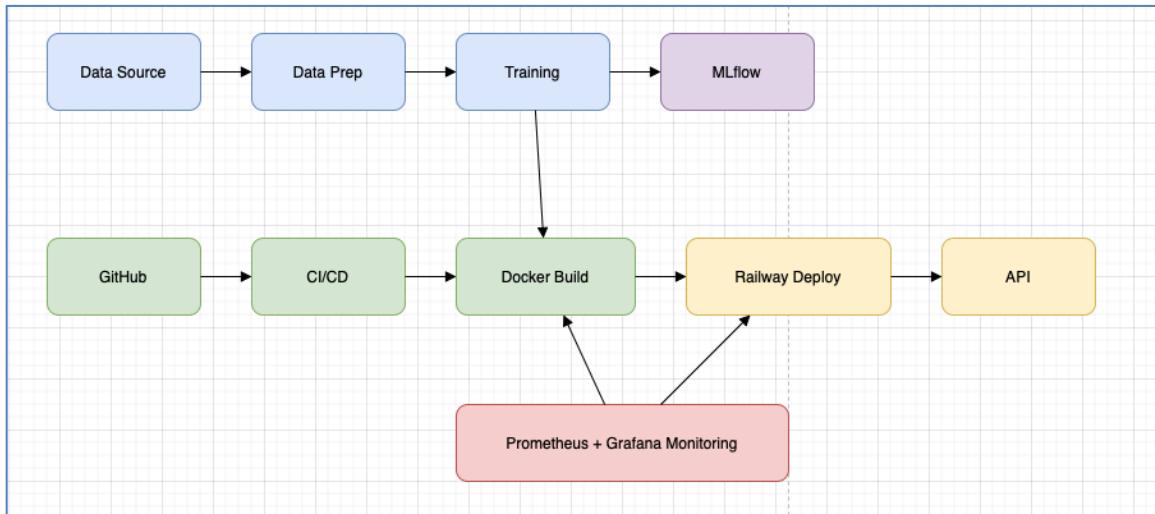
- Prometheus query for predict_requests_total



- 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

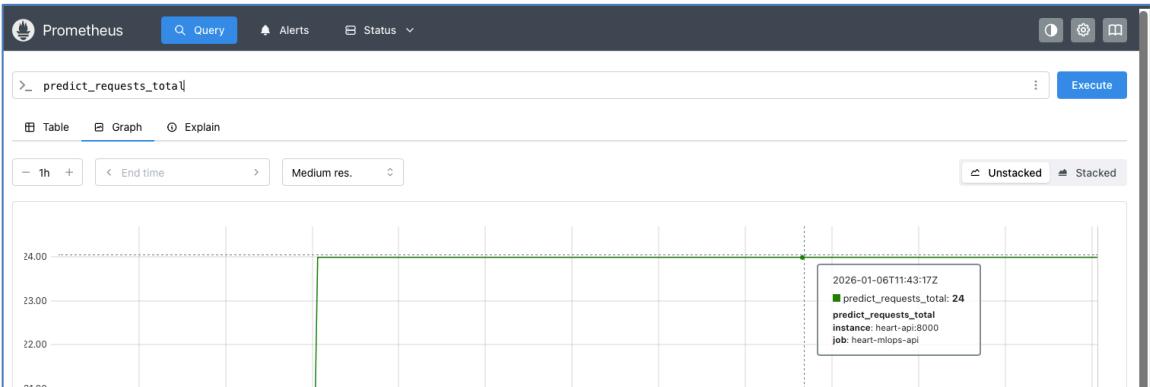
Manage environments

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:

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. Demo Video

[YouTube URL](#)

14. Repository Link

[GitHub Repository](#)

15. 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.