Heart Disease Prediction — End-to-End MLOps Case Study – Git URl **https://github.com/Jigyansh87/mlops-heart**

## Group No - 7

### Group Member Names:

1. Shivanshu Sharma - 2024ab05031 - 100% Contribution

2. Manish Kumar Tripathi - 2024aa05011 - 100% Contribution

3. HARSHIT KUMAR HALWAN - 2024aa05928 - 100% Contribution

4. JOHIT GARG - 2024aa05907 - 100% Contribution

5. JENNIFER SUSA SEN - 2024ac05319 - 100% Contribution

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 **https://github.com/Jigyansh87/mlops-heart** 2. cd mlops-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 >= 0.5 → 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 - 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. The system is reproducible, observable, and production-ready using industry-standard tools.