End-to-End MLOps Pipeline for Heart Disease Prediction

Group 100:

Abhijit Dey
Sukumati Naveen Kumar
Vinod Kumar A
Richa Singh
Gaurav Dhami

GitHub Repository:

https://github.com/2024aa05078-BITSAIML/BitsAIML2025_Mlops_Project

This project implements a complete end-to-end MLOps pipeline for heart disease prediction using clinical data. The objective is to demonstrate production-ready machine learning practices including data acquisition, exploratory data analysis (EDA), model training, experiment tracking, API-based inference, containerization, CI/CD automation, deployment, and monitoring.

The system is designed to be reproducible, testable, and deployable in an isolated environment using Docker and Kubernetes. This report documents the design decisions, implementation steps, and validation results.

1. Introduction

Machine Learning models often fail in real-world environments due to lack of reproducibility, monitoring, and deployment discipline. MLOps bridges this gap by combining machine learning with DevOps principles to ensure reliable, scalable, and maintainable ML systems.

In this project, we develop a heart disease prediction system using structured medical data and apply MLOps practices to:

- Ensure reproducible data processing and model training
- Package the model into a production-grade API
- Automate testing and validation
- Enable containerized deployment
- Provide observability through logging and monitoring

The outcome is a fully automated ML pipeline that can be executed from a clean environment with minimal setup.

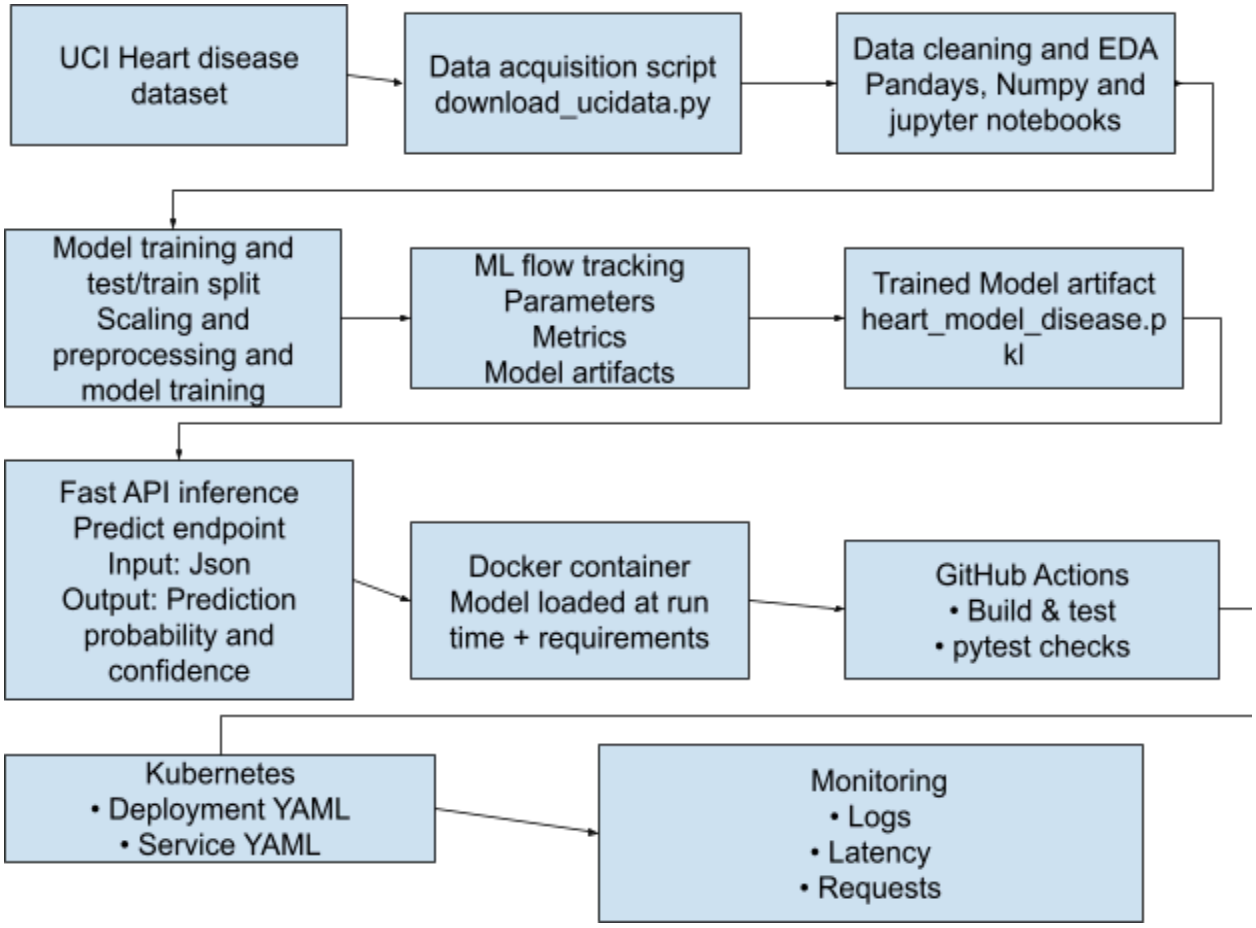
2. System Architecture

2.1 High-Level Architecture

The system architecture follows a modular MLOps design consisting of the following components:

1. **Data Layer** – Raw dataset acquisition and cleaning
2. **EDA & Feature Analysis** – Understanding data patterns
3. **Model Training Pipeline** – Training and saving the ML model
4. **Inference API** – FastAPI-based prediction service

5. **Containerization** – Docker-based isolated execution
6. **CI/CD Pipeline** – Automated testing and validation
7. **Deployment Layer** – Kubernetes-based deployment
8. **Monitoring & Logging** – Runtime observability



3. Setup & Installation Instructions

3.1 Prerequisites

- Python 3.9+
- Docker & Docker Desktop
- Git

- Kubernetes (Docker Desktop / Minikube)
- kubectl

3.2 Repository Setup

git clone https://github.com/2024aa05078-BITSAIML/BitsAIML2025_Mlops_Project.git
cd BitsAIML2025_Mlops_Project

3.3 Python Environment Setup

```
pip install -r requirements.txt
```

3.4 Docker Setup

```
docker build -t heart-disease-api .
docker run -p 8000:8000 heart-disease-api
```

Verify:

```
http://localhost:8000/docs
```

3.5 Kubernetes Deployment (Local)

```
kubectl apply -f deployment/kubernetes/k8s/
kubectl get pods
kubectl get services
```

4. Data Acquisition & Exploratory Data Analysis

4.1 Dataset Description

The dataset is sourced from the **UCI Machine Learning Repository** (Cleveland Heart Disease dataset). It contains 14 clinical attributes such as age, cholesterol

level, blood pressure, and exercise-induced angina, along with a binary target indicating presence or absence of heart disease.

4.2 Data Cleaning

The following preprocessing steps were applied:

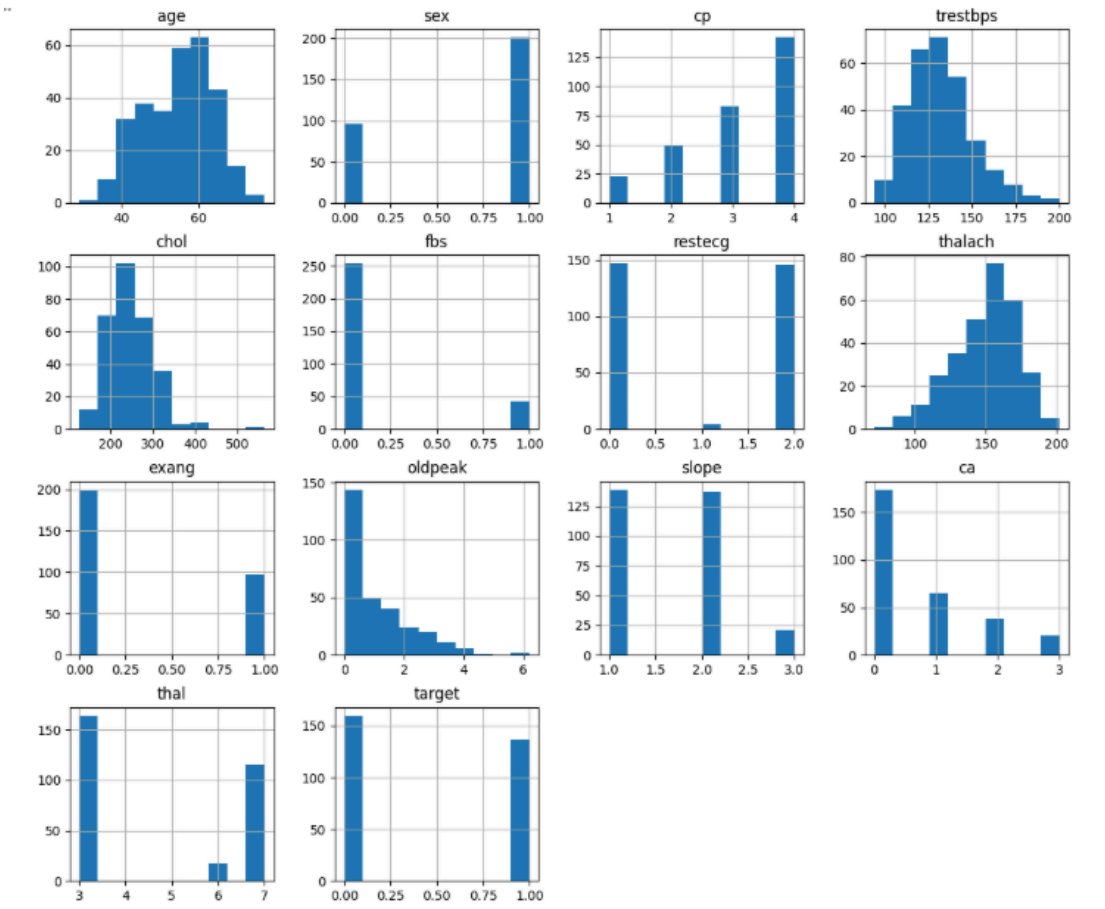
- Missing values marked as `?` were converted to NaN
- Rows with missing values were removed
- Target variable binarized:
    - `0` → No heart disease
    - `1` → Heart disease present

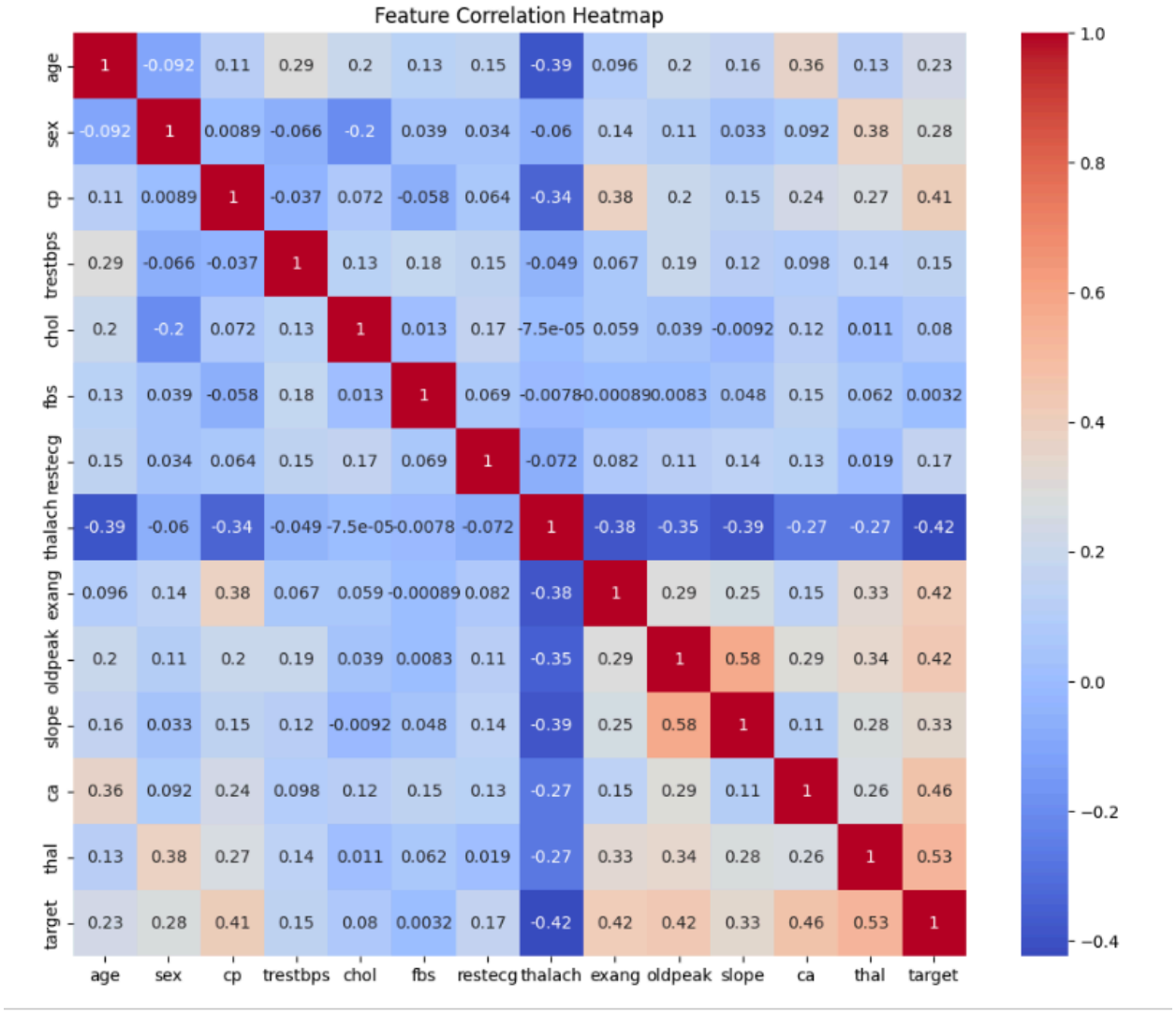
4.3 Exploratory Data Analysis

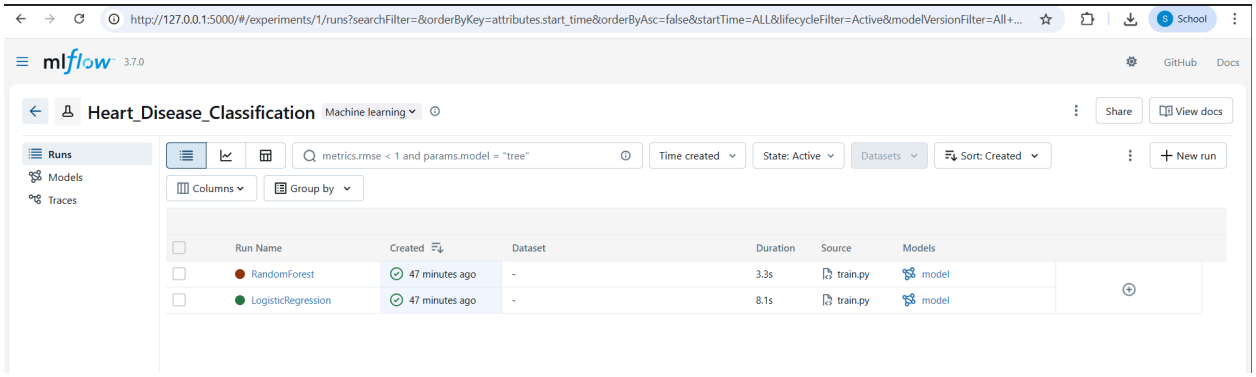
EDA was performed to understand feature distributions and relationships.

Key observations:

- Mild class imbalance in target variable
- Features such as `thalach`, `oldpeak`, and `exang` show strong correlation with heart disease
- No extreme outliers affecting model training


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target





5. Model Development & Training

5.1 Model Selection

A Logistic Regression model was selected due to:

- Interpretability
- Suitability for binary classification
- Stability with small-to-medium datasets

5.2 Feature Engineering & Training

- Features were scaled where required
- Train-test split was applied
- Model trained using scikit-learn
- Final trained model saved as a `.pkl` file

5.3 Experiment Tracking Summary

Although no external tracking tool (MLflow) was used, experiment tracking was ensured through:

- Versioned training scripts
- Saved model artifacts
- Logged metrics during training

Metrics evaluated:

- Accuracy
- Probability confidence score

6. Inference API Design

6.1 API Framework

FastAPI was used due to:

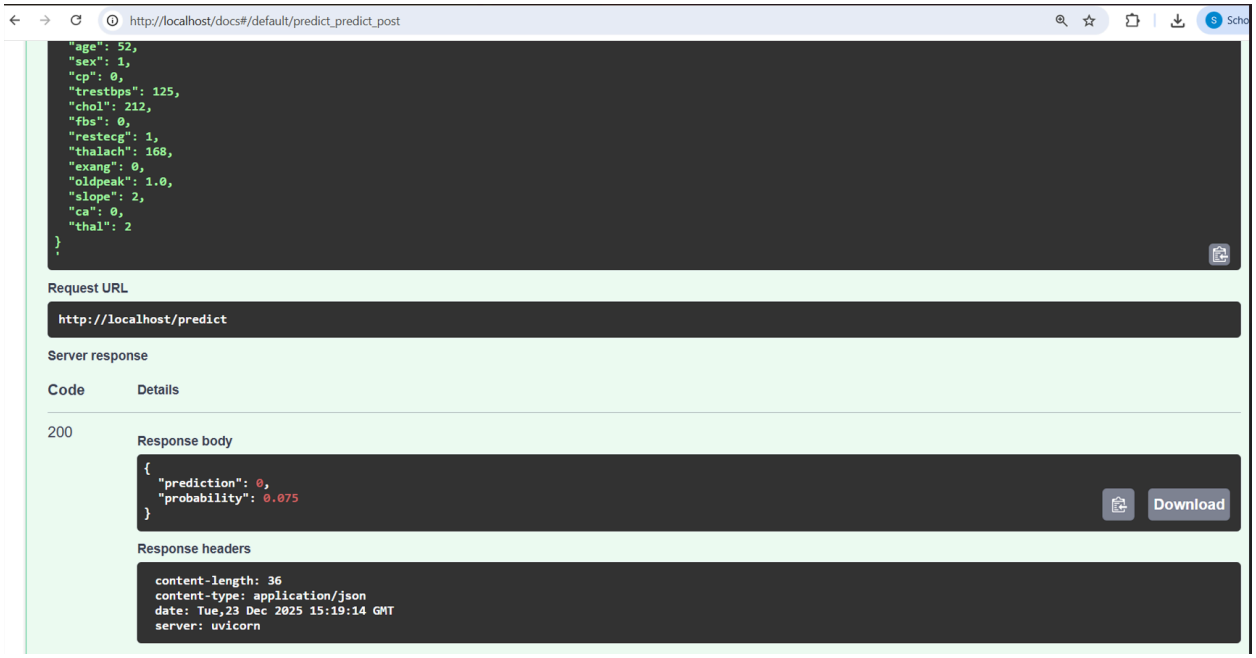
- Automatic Swagger documentation
- High performance
- Native request validation

6.2 API Endpoints

Endpoint	Method	Description
`/`	GET	Health check
`/predict`	POST	Heart disease prediction

Sample Response:

```
{
  "prediction": 0,
  "probability": 0.25
}
```



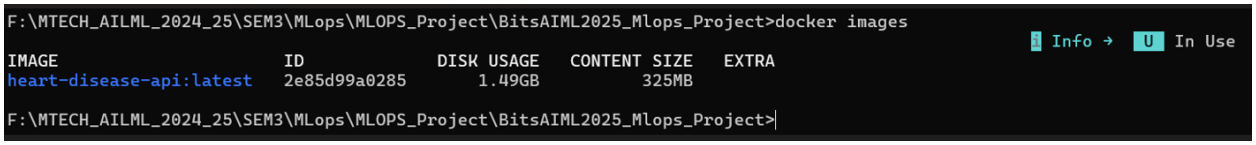
7. Containerization using Docker

The API and model are packaged into a Docker container to ensure:

- Environment consistency
- Portability
- Isolation from host system

Key Benefits

- One-command deployment
- No dependency conflicts
- Production-ready execution



```
F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project\deployment\kubernetes>docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
ed5f724adbe4   be41eefe6f90        "uvicorn src.app:app…"   5 minutes ago    Up 5 minutes                                                           k8s_heart-disease
-container_heart-disease-deployment-5bbbb78c4d-b7dpw_default_323c9b4b-d445-415e-b48e-ef8f8acfe9ab_0
d4db16f522c8   heart-disease-api   "uvicorn src.app:app…"   19 minutes ago   Up 19 minutes   0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp   flamboyant_jang

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project\deployment\kubernetes>docker logs ed5f724adbe4
/usr/local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator StandardScaler from version 1.8.0 when
using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator Pipeline from version 1.8.0 when using
version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator ColumnTransformer from version 1.8.0 wh
en using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator DecisionTreeClassifier from version 1.8
.0 when using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
/usr/local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator RandomForestClassifier from version 1.8
.0 when using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
2025-12-23 15:16:42,352 | INFO | Model loaded successfully
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     192.168.65.3:36572 - "GET /docs HTTP/1.1" 200 OK
INFO:     192.168.65.3:36572 - "GET /openapi.json HTTP/1.1" 200 OK
2025-12-23 15:19:15,935 | INFO | Received request: {'age': 52, 'sex': 1, 'cp': 0, 'trestbps': 125, 'chol': 212, 'fbs': 0, 'restecg': 1, 'thalach': 168, 'exa
ng': 0, 'oldpeak': 1.0, 'slope': 2, 'ca': 0, 'thal': 2}
2025-12-23 15:19:16,008 | INFO | Prediction successful | prediction=0, probability=0.075
INFO:     192.168.65.3:36356 - "POST /predict HTTP/1.1" 200 OK
```

8. CI/CD Pipeline

8.1 CI/CD Tooling

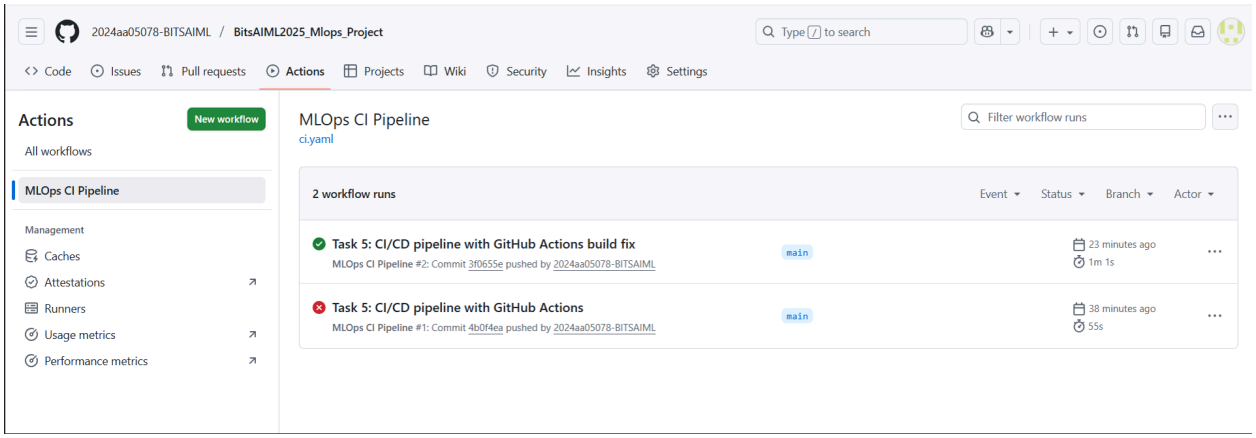
GitHub Actions was used to automate:

- Dependency installation
- Unit testing
- Pipeline failure on errors

8.2 Workflow Overview

On every push:

1. Code is checked out
2. Dependencies installed
3. Unit tests executed
4. Build fails if any test fails



9. Deployment & Monitoring

9.1 Kubernetes Deployment

- Deployment and Service YAML files used
- API deployed as a containerized pod
- Verified pod and service status using kubectl

9.2 Monitoring & Logging

- Request-level logging enabled
- Prediction count tracked
- Error logs generated on failures
- Logs accessible via container output

```
Microsoft Windows [Version 10.0.22631.6199]
(c) Microsoft Corporation. All rights reserved.

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project>kubectl apply -f deployment/kubernetes/deploy
ment.yaml
deployment.apps/heart-disease-deployment created

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project>kubectl apply -f deployment/kubernetes/servic
e.yaml
service/heart-disease-service created

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project>
```

```
Microsoft Windows [Version 10.0.22631.6199]
(c) Microsoft Corporation. All rights reserved.

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project>kubectl get nodes
NAME             STATUS   ROLES           AGE   VERSION
docker-desktop   Ready    control-plane   23h   v1.34.1

F:\MTECH_AILML_2024_25\SEM3\MLops\MLOPS_Project\BitsAIML2025_Mlops_Project>
```

10. Conclusion

This project successfully demonstrates a complete MLOps pipeline from data ingestion to production deployment. All components execute reliably in isolated environments, ensuring reproducibility and robustness.

The pipeline is production-ready and can be extended to cloud platforms such as AWS EKS or GCP GKE with minimal changes.