

Table of Contents

Group Info	3
Problem Statement.....	4
1. Setup and Installation Instructions.....	5
2. Data Acquisition and Exploratory Data Analysis	9
3. Experiment Tracking.....	11
4. Architecture Diagram.....	12
5. CI/CD Pipeline	13
6. Code Repository	15
7. Containerization and Deployment	15
8. Monitoring and Logging	15
9. Architecture Overview.....	17
10. Run using CURL/Postman API	18
11. Conclusion.....	19
Demo video	19

HEART DISEASE PREDICTION SYSTEM – MLOPS

Course	MLOps
Assignment	End-to-End MLOps Pipeline
Dataset	UCI Heart Disease Dataset
Group	Group 41
Repository	<u>https://github.com/rahulvg/MLOPS-Assignment-Group-41-</u>
Demo Video	<u>Link</u>

Group Info

Name	ID	Contribution
Abhay Kumar Mishra	2024ab05080	100
Bharatam Sai Sachin	2024AA05847	100
CHANDA RAKESH SANGANNA	2024aa05747	100
Galgali Rahul Vltharao	2024AB05079	100
Tiyas Chatterjee	2024AA05846	100

Problem Statement

The objective of this project is to design, develop, and deploy a scalable, reproducible, and production-ready machine learning system to predict the presence of heart disease based on patient health attributes.

The solution follows modern MLOps best practices, including experiment tracking, CI/CD automation, containerization, Kubernetes deployment, and monitoring.

1. Setup and Installation Instructions

1.1 Local Environment

Python Version: 3.10

Install Dependencies

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

Run Training File (Runs experiment with different parameter on Logistics Regression and Random Forest)

```
python train/train_experiment.py
```

Run all unit tests using Pytest

```
pytest
```

Create report using pytest

```
pytest --html=pytest_report.html
```

Launch MLflow UI (Local SQLite DB)

```
mlflow ui --backend-store-uri sqlite:///mlflow.db
```

Access MLflow at:

```
http://localhost:5000
```

1.2 Verification of Docker Build and Execution via GitHub Actions

Due to organizational restrictions that prevent local installation of Docker Desktop, the Docker image build and container execution were verified using **GitHub Actions**, which provides a Docker-enabled runner environment.

This ensures that containerization and execution are **reproducible, verifiable, and independent of local system constraints**.

1. Navigate to the GitHub repository:
<https://github.com/rahulvg/MLOPS-Assignment-Group-41->
2. Click on the **Actions** tab in the repository.
3. Select the most recent workflow run under the **CI pipeline**.
4. Open the workflow run and inspect the following steps:
 - **Build Docker image**
This step executes the Docker build command using the project's `Dockerfile`.
 - **Run Docker container and test API**
This step starts the container and invokes the `/predict` endpoint using a sample JSON request.

Evidence of Successful Docker Execution

Within the GitHub Actions workflow logs, the following evidence can be observed:

- Docker build logs confirming successful image creation
- Container startup logs indicating the FastAPI service is running
- Successful HTTP response from the `/predict` endpoint returning a prediction and confidence score

Screenshot of successful Docker run event

```
Run Docker container (smoke test)

1  ▶ Run docker run -d -p 8000:8000 heart-disease-api
15 61f71fe2a942f697dfc06582f8c8bbb8fab4b147568650d533ee8598021861f4
16 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
17                               Dload  Upload   Total   Spent    Left   Speed
18  0     0    0     0     0     0      0      0  --:--:-- --:--:-- --:--:--    0
19 100   92  100   33  100   59   4906   8771  --:--:-- --:--:-- --:--:-- 15333
20 {"prediction":1,"confidence":1.0}
```

1.3 Kubernetes (Local Deployment with Minikube)

Start Minikube

```
minikube start --container-runtime=containerd
```

```
Administrator: Windows PowerShell
PS E:\RGI3\MLOPS> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
heart-disease-api-9c4f666d8-mgxrw   1/1     Running   0           15s
PS E:\RGI3\MLOPS> kubectl get svc
NAME                                TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
heart-disease-service              NodePort    10.110.136.89 <none>        80:30007/TCP     15s
kubernetes                         ClusterIP   10.96.0.1     <none>        443/TCP          8m21s
PS E:\RGI3\MLOPS> minikube service heart-disease-service
```

NAMESPACE	NAME	TARGET PORT	URL
default	heart-disease-service	80	http://172.21.82.246:30007

```
* Opening service default/heart-disease-service in default browser...
PS E:\RGI3\MLOPS>
```

Inside Minikube

```
minikube image build -t heart-disease-api .
```

Deploy Application

```
kubectl apply -f k8s/deployment.yaml
```

```
kubectl apply -f k8s/service.yaml
```

Expose Service

```
minikube service heart-disease-service
```

[illegible]

2. Data Acquisition and Exploratory Data Analysis

2.1 Dataset

- Source: UCI Machine Learning Repository
- Format: CSV
- Task: Binary classification (presence or absence of heart disease)

2.2 Preprocessing

- Missing values handled
- Numerical features scaled using `StandardScaler`
- Target variable encoded
- Preprocessing implemented using a scikit-learn Pipeline

2.3 Exploratory Data Analysis (EDA) & Modelling choice

- Feature distributions analyzed using histograms
- Correlation heatmap used to study feature relationships
- Class balance verified

The modelling approach was guided by dataset characteristics, interpretability needs, and deployment stability.

Two models were evaluated:

- **Logistic Regression** – chosen as a strong, interpretable baseline for structured medical data
- **Random Forest** – included to capture non-linear relationships and feature interactions

All numerical features were standardized using **StandardScaler**, and preprocessing was implemented through a unified **scikit-learn Pipeline** to ensure reproducibility, prevent data leakage, and enable deployment-safe inference.

Hyperparameter Tuning

- Logistic Regression: $C \in \{0.1, 1.0, 10.0\}$
- Random Forest:
 - $n_estimators \in \{100, 200\}$
 - $max_depth \in \{None, 10\}$

Each configuration was logged as a separate experiment using **MLflow**.

Evaluation

Models were evaluated using **5-fold cross-validation** with the following metrics:

- Accuracy
- Precision
- Recall
- ROC-AUC

Final Model

Logistic Regression with $C = 0.1$ was selected due to:

- Consistent cross-validation performance
- Lower variance across folds
- Better generalization
- Simpler and more interpretable behavior

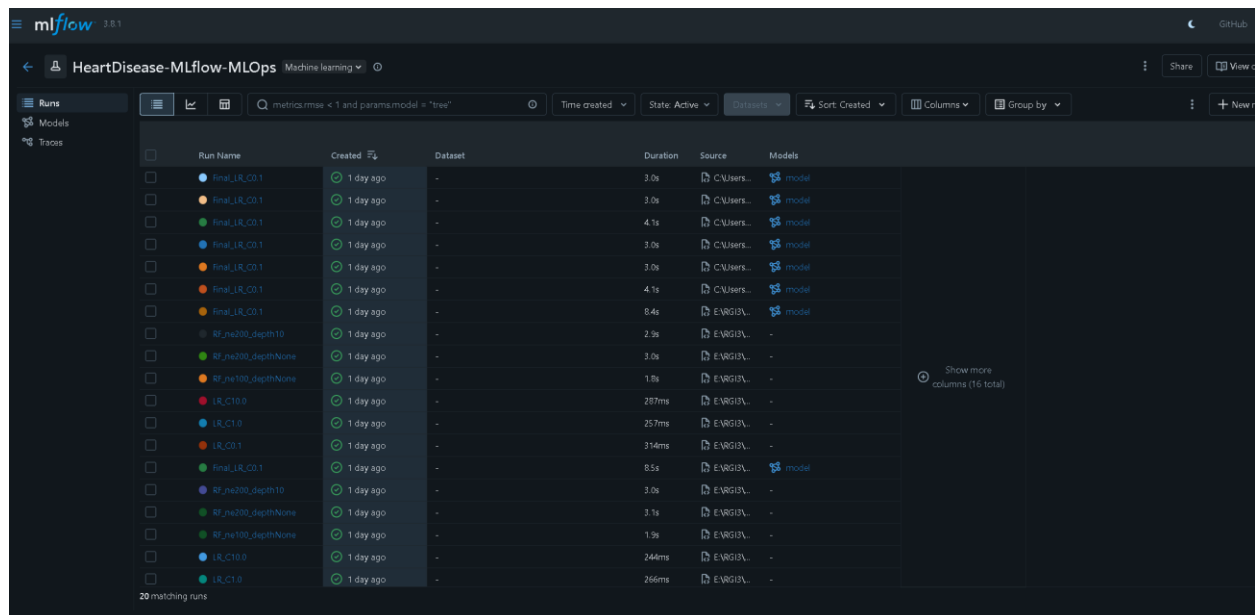
Its stability and ease of monitoring make it well-suited for a production-oriented MLOps pipeline.

3. Experiment Tracking

MLflow was integrated to track:

- Model parameters
- Cross-validation metrics
- Model artifacts

All experiments are logged under a dedicated MLflow experiment for easy comparison.

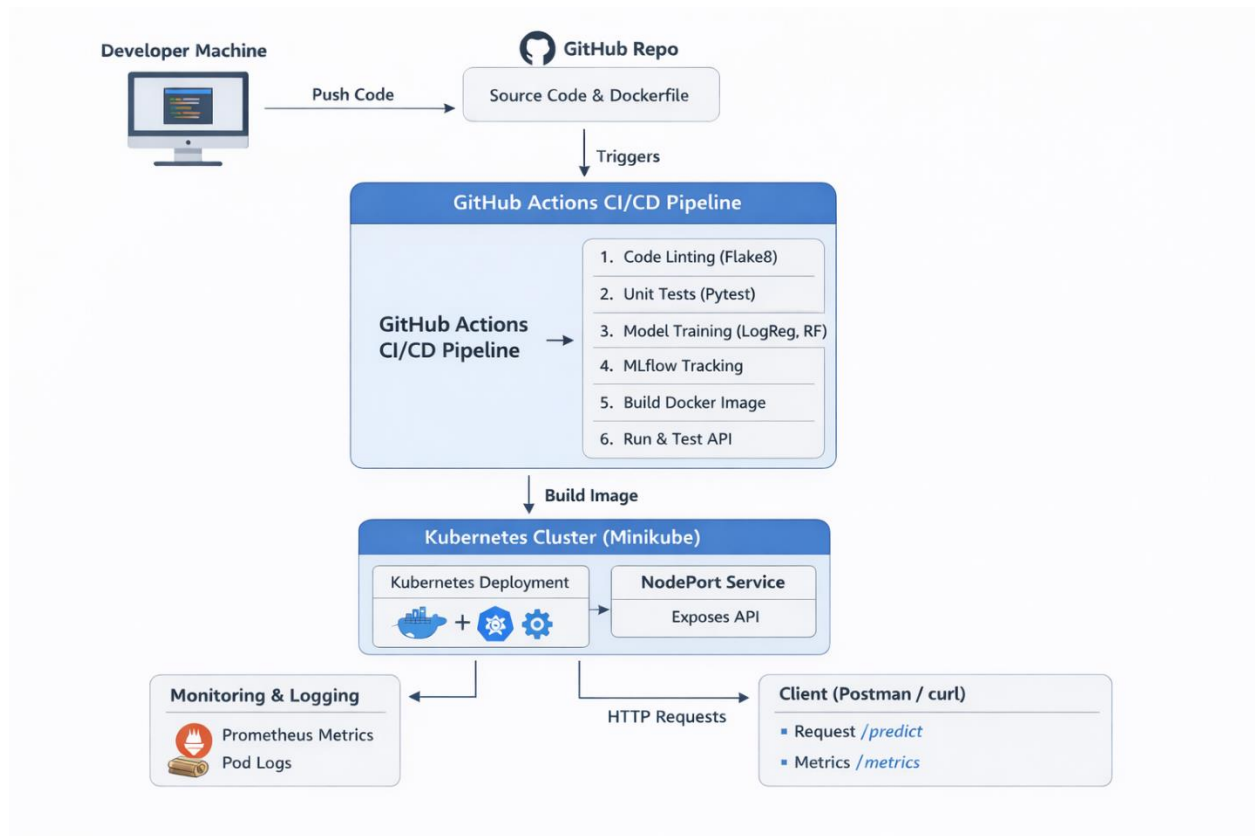


Run Name	Created	Dataset	Duration	Source	Models
final_lr_c01	1 day ago	-	3.0s	C:\Users...	model
final_lr_c01	1 day ago	-	3.0s	C:\Users...	model
final_lr_c01	1 day ago	-	4.1s	C:\Users...	model
final_lr_c01	1 day ago	-	3.0s	C:\Users...	model
final_lr_c01	1 day ago	-	3.0s	C:\Users...	model
final_lr_c01	1 day ago	-	4.1s	C:\Users...	model
final_lr_c01	1 day ago	-	8.4s	ENVIRON...	model
lr_c10.0	1 day ago	-	2.9s	ENVIRON...	-
lr_c10.0	1 day ago	-	3.0s	ENVIRON...	-
lr_c10.0	1 day ago	-	1.8s	ENVIRON...	-
lr_c10.0	1 day ago	-	287ms	ENVIRON...	-
lr_c10.0	1 day ago	-	257ms	ENVIRON...	-
lr_c10.0	1 day ago	-	314ms	ENVIRON...	-
final_lr_c01	1 day ago	-	8.5s	ENVIRON...	model
lr_c10.0	1 day ago	-	3.0s	ENVIRON...	-
lr_c10.0	1 day ago	-	3.1s	ENVIRON...	-
lr_c10.0	1 day ago	-	1.9s	ENVIRON...	-
lr_c10.0	1 day ago	-	244ms	ENVIRON...	-
lr_c10.0	1 day ago	-	266ms	ENVIRON...	-

Model Packaging and Reproducibility

- Final model saved as a serialized scikit-learn Pipeline
- Model can be found at **final_model\heart_disease_lr_c01.pkl** in git repo.
- Preprocessing included within the model
- Reproducible inference guaranteed
- Dependencies listed in requirements.txt
- Artifacts stored and versioned using MLflow check **mlflow_experiment.db** in git repo

4. Architecture Diagram



5. CI/CD Pipeline

Tools Used

- GitHub Actions
- Pytest
- Flake8
- Docker

Pipeline Stages

- Code linting
- Unit testing
- Model training
- Docker image build
- API smoke testing

The screenshot displays the GitHub Actions interface for a repository named 'MLOPS Assignment Group 41-'. The left sidebar shows the 'Actions' tab with a 'New workflow' button and a list of workflow runs. The main area shows 'All workflows' with a search bar and a table of workflow runs. The table lists 23 workflow runs, including Docker builds and a merge pull request.

Event	Status	Branch	Actor
✓ Docker	Success	mlops_assignment	Dec 31, 2025, 8:58 PM GMT+5:30
✓ Docker	Success	mlops_assignment	Dec 31, 2025, 6:42 PM GMT+5:30
✓ Docker	Success	mlops_assignment	Dec 31, 2025, 6:38 PM GMT+5:30
✓ Docker	Success	mlops_assignment	Dec 31, 2025, 6:20 PM GMT+5:30
✓ Merge pull request #5 from rahulvg/mlops_assignment	Success	main	Dec 30, 2025, 9:10 PM GMT+5:30
✓ Renamed requirement txt	Success	init-commit	Dec 30, 2025, 9:08 PM GMT+5:30

Summary

All Jobs

build test train

Run details

Usage

Workflow file

build-test-train

succeeded yesterday in 1m 44s

Search logs

Checkout code1s

Set up Python0s

Install dependencies34s

Run linting (flake8)1s

Run unit tests9s

Upload Pytest HTML report1s

Train final model9s

```
1  ▶ Run export PYTHONPATH=${pwd}
12 Registered model 'HeartDiseaseClassifier' already exists. Creating a new version of this model...
13 Created version '4' of model 'HeartDiseaseClassifier'.
14
15 ===== Logistic Regression Experiments =====
16
17 Run: LR_C0.1
18 ACCURACY | Mean: 0.8417 | Std: 0.0347
19 PRECISION | Mean: 0.8516 | Std: 0.0648
20 RECALL | Mean: 0.7987 | Std: 0.0277
21 ROC_AUC | Mean: 0.9144 | Std: 0.0213
22
23 Run: LR_C1.0
24 ACCURACY | Mean: 0.8482 | Std: 0.0382
25 PRECISION | Mean: 0.8679 | Std: 0.0506
26 RECALL | Mean: 0.7910 | Std: 0.0374
27 ROC_AUC | Mean: 0.9111 | Std: 0.0196
28
29 Run: LR_C10.0
30 ACCURACY | Mean: 0.8350 | Std: 0.0235
```

6. Code Repository

<https://github.com/rahulvg/MLOPS-Assignment-Group-41->

7. Containerization and Deployment

7.1 Dockerized API

- FastAPI-based service
- /predict endpoint
- Accepts JSON input
- Returns prediction and confidence score

7.2 Kubernetes Deployment

- Local Kubernetes using Minikube
 - Deployment and NodePort Service manifests
 - API tested using curl and Postman
-

8. Monitoring and Logging

8.1 Logging

- Request-level logging implemented via FastAPI middleware
- Logs include endpoint, HTTP status, and latency
- Logs accessible via Kubernetes pod logs

8.2 Monitoring

- Prometheus-compatible /metrics endpoint exposed
- Metrics include request count and request latency
- Ready for Prometheus and Grafana integration

Administrator: Windows PowerShell

```
heart-disease-api-9c4f666d8-fr998 1/1 Running 0 2m19s
PS E:\RG13\MLOPS> kubectl logs heart-disease-api-9c4f666d8-fr998
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator
:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator
r to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:348: InconsistentVersionWarning: Trying to unpickle estimator
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
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)
/usr/local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but S
warnings.warn(
/usr/local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarning: X does not have valid feature names, but S
warnings.warn(
2025-12-31 14:36:55,174 | INFO | POST /predict | status=200 | latency=0.003s
INFO: 10.244.0.1:44380 - POST /predict?content-type=application/json HTTP/1.1" 200 OK
PS E:\RG13\MLOPS>
```


9. Architecture Overview

Client (Postman / curl)

→ FastAPI API (/predict, /metrics)

→ Scikit-learn Pipeline

→ Kubernetes Pod

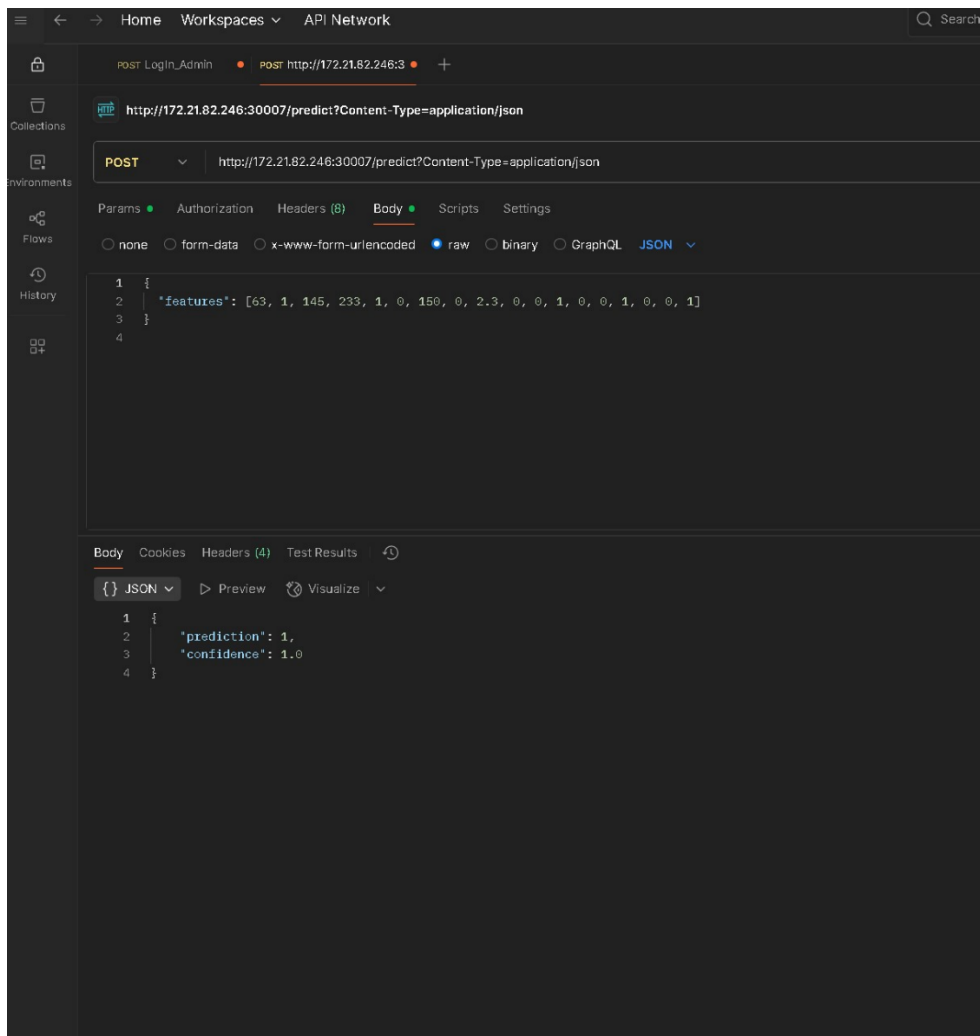
→ NodePort Service

CI/CD is handled using GitHub Actions, and experiment tracking is handled using MLflow.

10. Run using CURL/Postman API

Curl Command

```
"curl -X POST http://127.0.0.1:30007/predict \  
-H "Content-Type: application/json" \  
-d '{"features": [63,1,145,233,1,0,150,0,2.3,0,0,1,0,0,1,0,0,1]}'"
```



11. Conclusion

This project demonstrates a complete, production-grade MLOps workflow covering data analysis, model development, experiment tracking, CI/CD automation, containerization, Kubernetes deployment, and monitoring.

The system is scalable, reproducible, and aligned with real-world MLOps practices.

Demo video

[Google Drive Link for Demo Video](#)

[Github Link for demo video](#)