

MLOps Assignment I — End-to-End ML Model Development, CI/CD, and Production Deployment

1. Objective

Build a machine learning classifier for heart disease prediction and deploy it as a cloud-ready, monitored API using modern MLOps best practices.

2. Repository Link

- **GitHub:** github.com/chaitudevi/Heart-Disease-Classification-2

3. Dataset (Heart Disease UCI)

- **Source:** UCI Machine Learning Repository
- **Data acquisition:**
 - Script: `src/data/download_data.py`
 - Raw data folder: `data/raw/`

Screenshot(s)

```
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % python src/data/download_data.py
Dataset downloaded to /Users/darshan/Developer/repos/Heart-Disease-Classification-2/data/raw/heart_disease.csv
Column headers added successfully
```

4. Data Acquisition, Cleaning, and EDA (Task 1 — 5 marks)

4.1 Data Cleaning & Preprocessing

- Missing value handling, encoding, transformations
- Implementation: `src/data/preprocess.py`

4.2 EDA

- Notebook: `notebooks/01_eda.ipynb`
- Visuals: class balance, correlation heatmap, distributions

Screenshot(s)



5. Feature Engineering & Model Development (Task 2 — 8 marks)

5.1 Feature Pipeline

- Implementation: `src/features/feature_pipeline.py`

5.2 Models

- Logistic Regression
- Random Forest

5.3 Evaluation

- Cross-validation metrics: accuracy, precision, recall, ROC-AUC
- Training entrypoint: `src/models/train.py`

Screenshot(s)

```
darshan@Darshans-MacBook-Air ~ % python src/models/train.py
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/pydantic/_internal/_fields.py:149: UserWarning: Field "model_name" has conflict with protected namespace "model_"
.

You may be able to resolve this warning by setting `model_config['protected_namespaces'] = ()`.
    warnings.warn(
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/mlflow/tracking/_tracking_service/utils.py:178: FutureWarning: The filesystem tracking backend (e.g., './mlruns') will be deprecated in February 2026. Consider transitioning to a database backend (e.g., 'sqlite:///mlflow.db') to take advantage of the latest MLflow features. See https://github.com/mlflow/mlflow/issues/18534 for more details and migration guidance. For migrating existing data, https://github.com/mlflow/mlflow-export-import can be used.
    return FileStore(store_uri, store_uri)
[2026-01-03 13:54:13,265] INFO | src.features.feature_pipeline | Building full feature pipeline
[2026-01-03 13:54:13,265] INFO | src.features.feature_pipeline | Creating ColumnTransformer
[2026-01-03 13:54:13,265] INFO | src.features.feature_pipeline | Building numeric feature pipeline
[2026-01-03 13:54:13,265] INFO | src.features.feature_pipeline | Building categorical feature pipeline
[2026-01-03 13:54:13,268] INFO | src.features.feature_pipeline | Creating engineered features
[2026-01-03 13:54:13,277] INFO | src.features.feature_pipeline | Creating engineered features
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/sklearn/utils/extmath.py:203: RuntimeWarning: divide by zero encountered in matmul
    ret = a @ b
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/sklearn/utils/extmath.py:203: RuntimeWarning: overflow encountered in matmul
    ret = a @ b
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/sklearn/utils/extmath.py:203: RuntimeWarning: invalid value encountered in matmul
    ret = a @ b
[2026-01-03 13:54:13,282] INFO | src.features.feature_pipeline | Creating engineered features
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/sklearn/utils/extmath.py:203: RuntimeWarning: divide by zero encountered in matmul
    ret = a @ b
/Users/darshan/.pyenv/versions/3.10.18/lib/python3.10/site-packages/sklearn/utils/extmath.py:203: RuntimeWarning: divide by zero encountered in matmul
    ret = a @ b
[2026-01-03 13:54:13,543] INFO | src.features.feature_pipeline | Creating engineered features
[2026-01-03 13:54:13,559] INFO | src.features.feature_pipeline | Creating engineered features
[2026-01-03 13:54:13,576] INFO | src.features.feature_pipeline | Creating engineered features
[2026-01-03 13:54:13,662] INFO | src.features.feature_pipeline | Creating engineered features
[2026-01-03 13:54:13,680] INFO | src.features.feature_pipeline | Creating engineered features

Logistic Regression
accuracy: 0.8429
precision: 0.8620
recall: 0.7838
roc_auc: 0.9056

Random Forest
accuracy: 0.7933
precision: 0.8024
recall: 0.7297
roc_auc: 0.8853
[2026-01-03 13:54:13,700] INFO | src.features.feature_pipeline | Building full feature pipeline
[2026-01-03 13:54:13,700] INFO | src.features.feature_pipeline | Creating ColumnTransformer
```

```
r
[2026-01-03 13:54:13,700] INFO | src.features.feature_pipeline | Building numeric feature
pipeline
[2026-01-03 13:54:13,700] INFO | src.features.feature_pipeline | Building categorical feat
ure pipeline
[2026-01-03 13:54:13,701] INFO | src.features.feature_pipeline | Creating engineered featu
res
Best model selected: Logistic Regression
[2026-01-03 13:54:13,707] INFO | src.features.feature_pipeline | Creating engineered featu
res
```

6. Experiment Tracking (Task 3 — 5 marks)

- Tool: MLflow
- Logged:
 - Parameters
 - Metrics
 - Artifacts (ROC curve, confusion matrix, reports)

Screenshot(s)

The screenshot shows the MLflow interface at <http://localhost:5001/#/experiments>. The left sidebar has buttons for Home, Experiments (which is selected), Models, and Prompts. The main area is titled 'Experiments' and contains a table with one row. The table columns are Name, Time created, Last modified, Description, and Tags. The single experiment listed is 'Heart-Disease-Classification-2', created on 12/30/2025, 08:18:51 PM, last modified on 12/30/2025, 08:18:51 PM, with no description or tags.

Name	Time created	Last modified	Description	Tags
Heart-Disease-Classification-2	12/30/2025, 08:18:51 PM	12/30/2025, 08:18:51 PM	-	-

7. Model Packaging & Reproducibility (Task 4 — 7 marks)

- Model bundle saved at: `artifacts/model.pkl`
- Reproducible environment: `requirements.txt`
- Preprocessing/feature transformations are part of the saved sklearn `Pipeline`

Screenshot(s)

The image shows three screenshots of the mlflow interface:

- Runs Overview:** Shows a list of runs for the "Heart-Disease-Classification-2" experiment. The table includes columns for Run Name, Created, Dataset, Duration, Source, and Models. Three runs are listed: "Best_Model" (Logistic Regression), "Random Forest", and "Logistic Regression".
- Run Details:** Provides detailed information for the "Best_Model" run. It includes fields for Run ID, Name, Start Time, End Time, and Duration. It also shows parameters (model_type) and metrics (accuracy, precision, recall, roc_auc) for Logistic Regression and Random Forest.
- Artifacts View:** Displays the contents of the "MLmodel" artifact. The tree view shows "model", "metadata", and "MLmodel". The "MLmodel" node is expanded, showing its contents: conda.yaml, model.pkl, python_env.yaml, requirements.txt, model.pkl, and roc_curve.png. The right panel shows the YAML configuration for the "MLmodel" artifact, including fields like artifact_path, flavors, python_function, env, sklearn, and mlflow_version.

8. CI/CD Pipeline & Automated Testing (Task 5 — 8 marks)

- Tests under: [tests](#)
- CI workflow: [.github/workflows/github_actions.yaml](#)
- Stages: lint, test, train

Screenshot(s)

The screenshot shows the GitHub Actions CI Pipeline for a pull request. The pipeline summary indicates it was triggered via push last week by mlwithak, status is Success, total duration is 1m 33s, and there is 1 artifact. The pipeline consists of three stages: lint, test, and train, all of which have passed. The artifacts section shows a pytest-report file.

Name	Size	Digest
pytest-report	9.29 KB	sha256:8756ec0477f4660718db3c358b3a02832d8a...

The screenshot shows the test logs for the 'test' stage, which succeeded last week in 49s. The logs include numerous warnings and deprecation notices from matplotlib and pandas libraries, such as 'parseString' being deprecated in favor of 'parse_string'. The logs also mention 'pd.set_option('future.no_silent_downcasting', True)' and 'enablePackrat' being deprecated in favor of 'enable_packrat'.



9. Model Containerization (Task 6 — 5 marks)

- Dockerfile: [Dockerfile](#)

- API:
 - `/predict` accepts JSON, returns prediction + confidence
 - `/metrics` exposes Prometheus metrics

Screenshot(s)

```
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % docker build -t heart-disease-api:latest .
[+] Building 2.5s (11/11) FINISHED
  => [internal] load build definition from Dockerfile                               0.0s
  => => transferring dockerfile: 826B                                         0.0s
  => [internal] load metadata for docker.io/library/python:3.10-slim           2.3s
  => [internal] load .dockerignore                                              0.0s
  => => transferring context: 207B                                         0.0s
  => [1/6] FROM docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a 0.0s
  => => resolve docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a 0.0s
  => [internal] load build context                                              0.0s
  => => transferring context: 1.09MB                                         0.0s
  => CACHED [2/6] WORKDIR /app                                              0.0s
  => CACHED [3/6] RUN apt-get update && apt-get install -y --no-install-recommends 0.0s
  => CACHED [4/6] COPY requirements.txt ./requirements.txt                      0.0s
  => CACHED [5/6] RUN pip install --no-cache-dir -r requirements.txt          0.0s
  => [6/6] COPY . .                                                       0.0s
  => exporting to image                                                 0.1s
  => => exporting layers                                              0.0s
  => => exporting manifest sha256:9e558d454705301252952463e92faeee02e025a8eed006548a 0.0s
  => => exporting config sha256:351153723d5ba02edbd88e8fca055982324e9f1b23aff0b9ca9b 0.0s
  => => exporting attestation manifest sha256:658ff0f73f69b0ed51642dbe9168e6a75f20ec 0.0s
  => => exporting manifest list sha256:6bc4d960591420c677ee391fa46bd4a36fb4193421f89 0.0s
  => => naming to docker.io/library/heart-disease-api:latest                  0.0s
  => => unpacking to docker.io/library/heart-disease-api:latest                0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/p51azyqgbz6g5tu3rnlpv1e03
```

darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % docker run --rm -p 8000:8000 -v \$(pwd)/artifacts:/app/artifacts heart-disease-api:latest

```
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)
[2026-01-03 08:46:01,542] INFO | src.api.app | Handled GET /health -> 200 in 0.013s
INFO:     127.0.0.1:45314 - "GET /health HTTP/1.1" 200 OK
```

```
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % curl -X POST http://localhost:8000/predict -H "Content-Type: application/json" --data @sample_data/sample_request.json
{"prediction":0,"confidence":0.8095673592559162}
```

10. Production Deployment (Task 7 — 7 marks)

You deployed on **Docker Desktop Kubernetes** using manifests.

- Manifest: `k8s/deployment.yaml`
- Service exposure: LoadBalancer/NodePort

Screenshot(s)

```
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % kubectl apply -f k8s/deployment.yaml
deployment.apps/heart-disease-api unchanged
service/heart-disease-api unchanged
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 %

darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % kubectl get svc heart-disease-api
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
heart-disease-api   LoadBalancer  10.103.110.219  localhost   80:31677/TCP  3d2h
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 %

darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 % kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
heart-disease-api-867c4f967b-wb2xl  1/1     Running   0          3d2h
darshan@Darshans-MacBook-Air Heart-Disease-Classification-2 %
```

The screenshot shows the Docker Desktop Kubernetes interface. At the top, there's a navigation bar with icons for recent projects, Docker Desktop logo, and personal settings. Below the navigation bar, the main area is divided into several sections:

- Cluster:** Shows one node (1) running Kubernetes version v1.34.1.
- Deployments:** Shows a single deployment named "heart-disease-api" in the "default" namespace, status: Available, with 1/1 pods.
- Pods:** Shows a single pod named "heart-disease-api-867c4f967b-wb2xl" in the "default" namespace, status: Running.
- Nodes:** Shows one node named "docker-desktop" in the "default" namespace, status: Ready.
- Services:** Shows two services: "heart-disease-api" with Cluster IP 10.103.110.219 and port 80/TCP, and "kubernetes" with Cluster IP 10.96.0.1 and port 443/TCP.

11. Monitoring & Logging (Task 8 — 3 marks)

11.1 API Logging

- Middleware request logging in: `src/api/app.py`

11.2 Metrics + Dashboards

- Metrics endpoint: `/metrics`
- Prometheus config: `monitoring/prometheus.yml`
- Grafana provisioning + dashboard:
 - `monitoring/grafana/provisioning/`
 - `monitoring/grafana/dashboards/heart-disease-api-dashboard.json`

Screenshot(s)

Screenshot of the Prometheus web interface showing target health for the 'heart-disease-api' job. The endpoint is `http://host.docker.internal:80/metrics` and it is marked as UP.

Screenshot of Grafana displaying various metrics for the Heart Disease API. The top section shows Request rate (req/s), Latency p50/p95/p99 (seconds), HTTP error rate (4xx/5xx req/s), and Prediction confidence p50. The bottom section shows Top 5 endpoints by request rate and Latency p95 by endpoint (seconds). A terminal window at the bottom shows log output from the application.

```
[2026-01-03 08:51:34,141] INFO | src.api.app | Handled GET /metrics -> 200 in 0.003s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
[2026-01-03 08:51:37,364] INFO | src.api.app | Handled GET /health -> 200 in 0.000s
INFO: 10.1.0.1:65136 - "GET /health HTTP/1.1" 200 OK
[2026-01-03 08:51:39,130] INFO | src.api.app | Handled GET /metrics -> 200 in 0.003s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
[2026-01-03 08:51:44,132] INFO | src.api.app | Handled GET /metrics -> 200 in 0.003s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
[2026-01-03 08:51:47,365] INFO | src.api.app | Handled GET /health -> 200 in 0.001s
INFO: 10.1.0.1:62480 - "GET /health HTTP/1.1" 200 OK
[2026-01-03 08:51:49,133] INFO | src.api.app | Handled GET /metrics -> 200 in 0.003s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
[2026-01-03 08:51:51,582] INFO | src.api.app | Handled GET /health -> 200 in 0.001s
INFO: 10.1.0.1:61260 - "GET /health HTTP/1.1" 200 OK
[2026-01-03 08:51:54,134] INFO | src.api.app | Handled GET /metrics -> 200 in 0.003s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
[2026-01-03 08:51:57,367] INFO | src.api.app | Handled GET /health -> 200 in 0.001s
INFO: 10.1.0.1:61274 - "GET /health HTTP/1.1" 200 OK
[2026-01-03 08:51:59,132] INFO | src.api.app | Handled GET /metrics -> 200 in 0.004s
INFO: 192.168.65.3:55982 - "GET /metrics HTTP/1.1" 200 OK
```

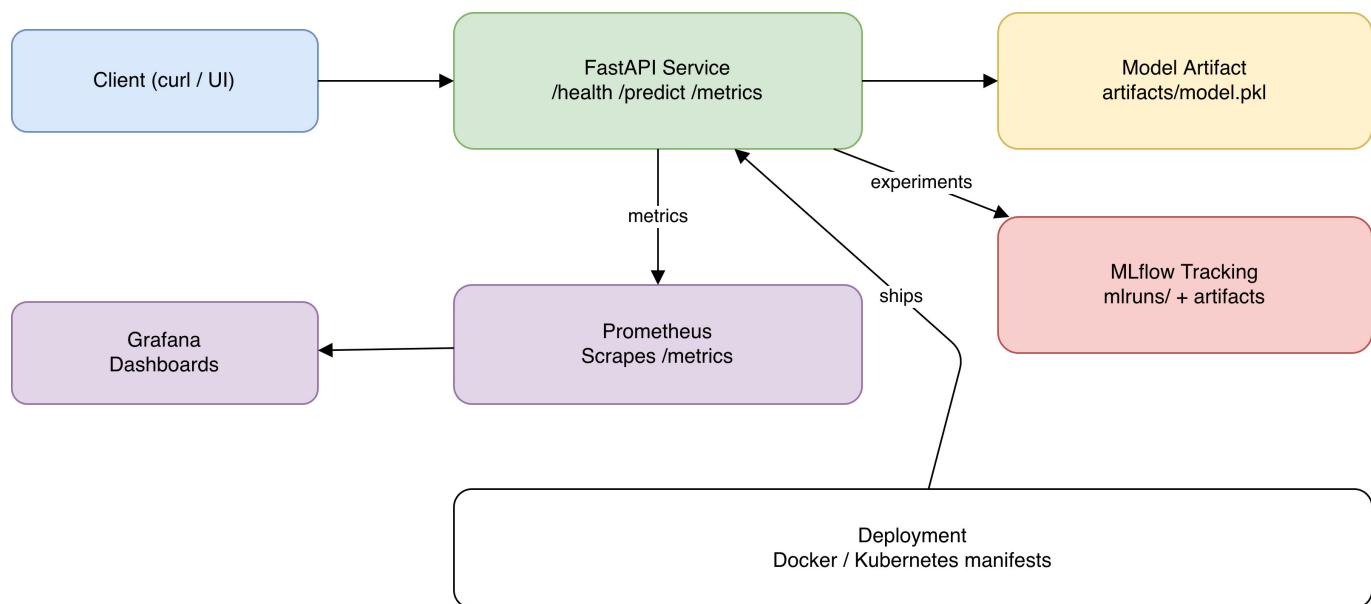
12. System Architecture Diagram

- Draw.io XML: [reports/system_architecture.xml](#)

- Architecture flow:

```
graph TD
    A[Client / UI] --> B[FastAPI Service]
    B --> C[Model Artifact joblib]
    B --> D[Prometheus Metrics]
    B --> E[Logging]
    E --> F[Stdout / Aggregator]
    D --> G[Grafana / Prometheus]
```

Screenshot(s)



13. How to Reproduce (Production-Readiness)

13.1 From clean environment

```
# 1. Clone and setup
git clone https://github.com/chaitudevi/Heart-Disease-Classification-2
cd Heart-Disease-Classification-2
python -m venv venv && source venv/bin/activate
pip install -r requirements.txt

# 2. Data pipeline
python src/data/download_data.py
python src/data/load_data.py

# 3. Train model (generates artifacts/model.pkl)
python src/models/train.py

# 4. Run API locally
uvicorn src.api.app:app --host 0.0.0.0 --port 8000

# 5. Test API
```

```
curl -X POST http://localhost:8000/predict -H "Content-Type: application/json" \
-d @sample_data/sample_request.json

# 6. Run tests
pytest tests/ -v

# 7. Build container
docker build -t heart-disease-api .

# 8. Deploy to Kubernetes
kubectl apply -f k8s/deployment.yaml

# 9. Start monitoring stack
docker-compose -f docker-compose.monitoring.yml up -d

# 10. View MLflow experiments
mlflow ui --host 0.0.0.0 --port 5000
```

13.2 Key Config Files

- `monitoring/prometheus.yml`: Prometheus scrape config
- `monitoring/grafana/provisioning/`: Grafana auto-provision
- `k8s/deployment.yaml`: Kubernetes manifests
- `.github/workflows/github_actions.yaml`: CI/CD pipeline
- `src/models/train.py`: Training script with MLflow logging
- `src/api/app.py`: FastAPI app with metrics and logging

13.3 Ports & Endpoints

- API: `http://localhost:8000` (or k8s service IP)
- Health: `http://localhost:8000/health`
- Metrics: `http://localhost:8000/metrics`
- Prometheus: `http://localhost:9090`
- Grafana: `http://localhost:3000` (admin/admin123)
- MLflow: `http://localhost:5000`

14. Appendix: Evidence Checklist

- CI run screenshots
- Deployment screenshots
- Grafana dashboard screenshot
- MLflow runs screenshot