BITS Pilani
Pilani | Dubai | Goa | Hyderabad

MLOps ((S1-25_AIMLCZG523)
ASSIGNMENT - I

Total Marks-50

MLOps Experimental Learning Assignment: End-to-End ML Model Development, CI/CD, and Production Deployment Experimental Learning

Objective:

Design, develop, and deploy a scalable and reproducible machine learning solution utilising modern MLOps best practices. The assignment emphasises practical automation, experiment tracking, CI/CD pipelines, containerization, cloud deployment, and monitoring—mirroring real-world production scenarios.

Dataset: Title: Heart Disease UCI Dataset
Source: UCI Machine Learning Repository

- CSV containing 14+ features (age, sex, blood pressure, cholesterol, etc.) and a binary target (presence/absence of heart disease).

Problem Statement: Build a machine learning classifier to predict the risk of heart disease based on patient health data, and deploy the solution as a cloud-ready, monitored API.

Assignment Tasks

1. **Data Acquisition & Exploratory Data Analysis (EDA)** **[5 marks]**

Obtain the dataset (provide download script or instructions). Clean and preprocess the data (handle missing values, encode features). Perform EDA with professional visualizations (histograms, correlation heatmaps, class balance).

2. Feature Engineering & Model Development **[8 marks]**

Prepare the final ML features (scaling and encoding). Build and train at least two classification models (e.g., Logistic Regression and Random Forest). Document model selection and tuning process. Evaluate using cross-validation and relevant metrics (accuracy, precision, recall, ROC-AUC).

3. Experiment Tracking **[5 marks]**

Integrate MLflow (or a similar tool) for experiment tracking. Log parameters, metrics, artifacts, and plots for all runs.

4. Model Packaging & Reproducibility **[7 marks]**

Save the final model in a reusable format (e.g., MLflow, pickle, ONNX). Write a clean requirements.txt (or Conda env file). Provide a preprocessing pipeline/transformers to ensure full reproducibility.

5. CI/CD Pipeline & Automated Testing **[8 marks]**

Write unit tests for data processing and model code (Pytest or unit test). Create a GitHub Actions (or Jenkins) pipeline That Includes Linting, unit testing, and model training steps. Artifacts/logging for each workflow run.

6. Model Containerization **[5 marks]**

Build a Docker container for the model-serving API (Flask or FastAPI is recommended). Expose /predict endpoint, accept JSON input, return prediction and confidence. The container must be built and run locally with the sample input.

7. Production Deployment **[7 marks]**

Deploy the Dockerized API to a public cloud or local Kubernetes (GKE, EKS, AKS, or Minikube/Docker Desktop).

Use a deployment manifest or Helm chart. Expose via Load Balancer or Ingress. Verify endpoints and provide deployment screenshots.

8. Monitoring & Logging **[3 marks]**

Integrate logging of API requests. Demonstrate simple monitoring (Prometheus + Grafana or API metrics/logs dashboard).

9. Documentation & Reporting **[2 marks]**

Submit a professional Markdown or PDF report including:

- Setup/install instructions.
- EDA and modelling choices.
- Experiment tracking summary.
- Architecture diagram.
- CI/CD and deployment workflow screenshots.
- Link to code repository.

Deliverables

a) GitHub repository with:
   - Code, Dockerfile(s), requirements.txt/env.yml
   - Cleaned dataset and download script/instructions
   - Jupyter notebooks/scripts (EDA, training, inference)
   - test/ folder with unit tests
   - GitHub Actions workflow YAML (or Jenkinsfile)
   - Deployment manifests/Helm charts
   - Screenshot folder for reporting
   - Final written report 10 pages as a doc/docx file.

b) Short video containing an end-to-end pipeline

c) Deployed API URL (if public) or access instructions (for local testing)

Production-Readiness Requirements

- All scripts must execute from a clean setup using the requirements file.
- Model must serve correctly in an isolated environment (Docker; container build/test proof required).
- Pipeline must fail on code or test errors and give clear logs.