



MLOps ((S1-25_AIMLCZG523))

Assignment 2 [Total Marks – 50]

Design and implement an end-to-end MLOps pipeline for model building, artifact/image creation, packaging, containerization, and CI/CD-based deployment using open-source tools

Use case : Binary image classification (Cats vs Dogs) for a pet adoption platform.

Dataset : Cats and Dogs classification dataset

[CATS and Dogs binary classification dataset from Kaggle](#)

Pre-process to 224x224 RGB images for standard CNNs

Split into train/validation/test sets (e.g., 80%/10%/10%). Use data augmentation for better generalization

M1: Model Development & Experiment Tracking -

10M

Objective: Build a baseline model, track experiments, and version all artifacts.

Tasks:

1. Data & Code Versioning

Use Git for source code versioning (project structure, scripts, and notebooks).

Use DVC (or Git-LFS) for dataset versioning and to track pre-processed data.

2. Model Building

Implement at least one baseline model (e.g., simple CNN or logistic regression on flattened pixels).

Save the trained model in a standard serialized format (e.g., .pkl, .pt, .h5).

3. Experiment Tracking

Use an open-source tracker like MLflow/Neptune to log runs, parameters, metrics, and artifacts (confusion matrix, loss curves)

M2: Model Packaging & Containerization –

10M

Objective: Package the trained model into a reproducible, containerized service.

Tasks:

1. Inference Service

Wrap the trained model with a simple REST API using FastAPI/Flask.

Implement at least two endpoints: health check and prediction (accepts input and returns class probabilities/label).

2. Environment Specification

Define dependencies using requirements.txt

Ensure version pinning for all key ML libraries for reproducibility.

3. Containerization

Create a Dockerfile to containerize the inference service.

Build and run the image locally and verify predictions via curl/Postman

M3: CI Pipeline for Build, Test & Image Creation –

10M

Objective: Implement Continuous Integration to automatically test, package, and build container images

Tasks:

- Automated Testing :** Write unit tests for at least one data pre-processing function and One model utility/inference function. Ensure tests run via pytest or similar.



2. CI Setup (Choose one: GitHub Actions / GitLab CI / Jenkins / Tekton)

Define a pipeline that on every push/merge request, checks out the repository, installs dependencies, runs unit tests, and builds the Docker image

3. Artifact Publishing:

Configure the pipeline to push the Docker image to a container registry (e.g., Docker Hub, GitHub Container Registry, local registry).

M4: CD Pipeline & Deployment –

10M

Objective: Implement Continuous Deployment of the containerized model to a target environment.

Tasks:

1. Deployment Target

Choose one: local Kubernetes cluster (kind/minikube/microk8s), Docker Compose, or a simple VM server.

Define infrastructure manifests: For Kubernetes: Deployment + Service YAML.
For Docker Compose: docker-compose.yml.

2. CD / GitOps Flow

Extend CI or use a CD tool (Argo CD, Jenkins, GitHub Actions environment) to:

- Pull the new image from the registry.
- Deploy/update the running service automatically on main branch changes.

3. Smoke Tests / Health Check

Implement a simple post-deploy smoke test (e.g., script that calls the health endpoint and one prediction call).

Fail the pipeline if smoke tests fail

M5: Monitoring, Logs & Final Submission –

10M

Objective: Monitor the deployed model and submit a consolidated package of all artifacts.

Tasks:

1. Basic Monitoring & Logging

Enable request/response logging in the inference service (excluding sensitive data).

Track basic metrics such as request count and latency (via logs, Prometheus, or simple in-app counters).

2. Model Performance Tracking (Post-Deployment)

Collect a small batch of real or simulated requests and true labels.

Deliverables

1. A zip file containing all source code, Configuration files (DVC, CI/CD, Docker, deployment manifests), and Trained model artefacts (If video file too large, Kindly share the link)
2. Screen recording of less than 5 minutes demonstrating the complete MLOps workflow from code change to deployed model prediction