Group 40:

2023aa05995	**Akshara.B**
2023ab05053	**Chandhira sekaran N**
2023aa05501	**Deepak C C**
2023aa05996	**Rajesh J**

Description of the work done.

M1: Set Up a CI/CD Pipeline & Version Control

- Configured a CI/CD pipeline using GitHub Actions for a Sample ML project.
- The pipeline included the following key stages:
    - Checkout Code, Set Up Python, Install Dependencies, Linting, Train Model, Verify Model, Run Tests, Deployment.
- Designed the pipeline to trigger on events like Push and Pull request.
- Managed the project using **Git** and hosted it on **GitHub**.
- Demonstrated proper branching and merging workflows:
    - Created feature branches for modular development.
    - Opened PR for code review and merged approved changes into main branch.
- Maintained clear commit messages and a well-structured repository.

Justifications for M1:

GitHub for Version Control: Easy to use, fosters collaboration, and widely adopted for seamless project sharing.
GitHub Actions for CI/CD: Integrates with repositories, automates workflows, offers customizable YAML pipelines, and is cost-effective.
Pipeline Design: Focused on linting, testing, and deployment to ensure quality, functionality, and end-to-end automation.

M2: Process and Tooling

- The Boston Housing dataset was fetched, processed, and saved as *boston_housing.csv*.
- **Experiment Tracking with MLflow**: Four Random Forest models with varying n_estimators were trained. Parameters, MSE metrics, and models were logged in MLflow for comparison and reproducibility.
- **Data Versioning with DVC**: The dataset was versioned using DVC, enabling tracking of changes and reversion to previous states. Git was used for DVC-tracked files.
- **Automation**: A Python script automated data fetching, DVC setup, and MLflow experiment tracking.

Justifications for M2:

Dataset: The Boston Housing dataset is small and well-suited for regression tasks.
MLflow: Simplifies logging and comparison of experiments, ensuring reproducibility.
DVC: Efficiently manages dataset versioning alongside code with Git.
Random Forest: An ensemble model for exploring hyperparameter effects (n_estimators).
Automation: Streamlines the workflow for consistency and efficiency.

M3: Model Experimentation and Packaging

- The Iris Dataset and RandomForestClassifier model was selected within sklearn.
- Hyperparameters namely n_estimators, max_depth, min_samples_split were used.

- **Obtuna** used for HP Tuning for 50 trials and the best attempt achieved in 2nd trial.
- Best hyperparameters used to prepare and train model & exported using joblib.
- Exported model used within a flask app and then to Docker Images/Containers.

<u>Justifications for M3:</u>

Iris dataset and **RandomForestClassifier** were simple and more preferred for selection.
Obtuna is a very powerful tool for Hyperparameter Tuning with very high flexibility.
Flask framework supports easy and fast creation of Web API to serve the trained Model.
Docker is used for easy Containerization of the environment and creating images.

M4: Model Deployment and Orchestration

- AWS was used for the Cloud provider which supports **Sagemaker AI** for easy training compute instances like JupyterLab.
- Elastic Container Registry used for Image Registry
- EKS – Elastic Kubernetes Service for deploying using Kubernetes.

<u>Justifications for M4:</u>

AWS is the most widely adopted and provides complete solutions for Cloud
Sagemaker AI is an AWS offering for easy training and deployment for AI models.
Elastic Container Registry for storing the docker/container images.
EKS – Managed Kubernetes solutions offering from AWS.