Description of the Work Completed

This project builds an MLOps pipeline integrating tools like Optuna for optimization, MLflow for experiment tracking, DVC for version control, and Flask for API deployment. The application is containerized with Docker and deployed to GKE using Kubernetes and Helm charts for efficient management and scaling.

M1: Initial Setup and CICD Pipeline

CICD Pipeline: Implemented a Continuous Integration and Continuous Deployment (CICD) pipeline to automate the testing and deployment process.
Report: Detailed documentation of the CICD pipeline is available here.

M2: Experimentation and Version Control

Mlflow Experiment: Conducted experiments using MLflow to track and manage machine learning models.
DVC Repo: Data Version Control (DVC) was used to manage datasets and model versions.
Report: The process and tooling are documented here.

M3: Model Tuning and Containerization

Hyperparameter Tuning: Performed hyperparameter tuning using the script python hyper_parameter.py. The results were documented.
Dockerization: Containerized the application using Docker. The application can be run in docker container using:
docker run -p 5050:5050 iris-pred-flask-app
Report: Detailed documentation of the model experimentation and packaging is available here.

M4: Deployment and Orchestration

Deployed Model Endpoint: The model was deployed to Google Kubernetes Engine (GKE) and can be accessed here.
Kubernetes Files: Kubernetes (K8) files for deployment are available here.
Deployment and Orchestration Process: The deployment and orchestration process is documented here.

Justification for the Choices Made

CICD Pipeline: Automating the testing and deployment process ensures that the code is always in a deployable state, reducing the risk of integration issues and speeding up the development cycle.

MLflow for Experiment Tracking: MLflow provides a robust framework for tracking experiments, which is crucial for reproducibility and collaboration in machine learning projects.

DVC for Version Control: DVC allows for efficient management of datasets and model versions, ensuring that the team can easily track changes and revert to previous versions if necessary.

Optuna: Optuna is an automatic hyperparameter optimization software framework, particularly useful for tuning machine learning models. It provides a flexible and efficient way to find the best hyperparameters, which can significantly improve model performance. Its ease of integration with scikit-learn and other machine learning libraries makes it a valuable tool for model optimization.

Flask: Flask is a lightweight WSGI web application framework in Python. It is designed with simplicity and flexibility in mind, making it easy to set up and run web applications. Flask is ideal for deploying machine learning models as RESTful APIs, allowing for easy integration with other services and applications.

Docker for Containerization: Docker ensures that the application runs consistently across different environments, simplifying the deployment process and improving scalability.

Kubernetes for Orchestration: Kubernetes provides a powerful platform for automating deployment, scaling, and management of containerized applications, ensuring high availability and efficient resource utilization.

Google Kubernetes Engine (GKE): GKE provides a managed Kubernetes service for deploying and managing containerized applications in Google Cloud Platforms (GCP).

These choices collectively enhance the efficiency, reproducibility, and scalability of the machine learning workflow, aligning with best practices in MLOps.