# 

# **Final Deliverables**

## 

## **Group Members Name with Student ID:**

* **A BIJU - 2023AA05553**
* **DIVYA S - 2023AA05529**
* **MATHEW KADAMBATT - 2023AA05887**
* **NIKHIL N B - 2023AA05824**

## **Description of the work completed**

### **M1: MLops Foundations**

* Created a new repository in Git
* Showcased the branching and merging using dev branch, deploy branch, and main branch.
* Created CI Pipeline using github actions with following stages
  + Linting using flake8
  + Testing using pytest (Data preprocessing testing, model evaluation testing, prediction testing)
  + Build-and-push (Dockerize the application and push to docker repository
  + Deploy (Deploy the dockerized image to EKS cluster)

### **M2: Process and Tooling**

* Experiment tracking using MLFlow
  + Store the model to Mlflow
  + Store the best hyperparameters in mlflow
  + Store evaluation metrics in mlflow
  + Download the latest model from MLFlow for prediction
* DVC
  + Dvc init
  + Commit different datasets to DVC
  + Show case rollback to different versions

### **M3: Model Experimentation and Packaging**

* Created a model for Diabetic prediction
  + Dataset: National Institute of Diabetes and Digestive and Kidney Diseases
* Hyperparameter tuning using 4 parameters with total 36 combinations.
  + Using GridSearchCV
  + Package the best model as pickle file
  + Push the best model to MLFlow
* Provide API for prediction using the best models found
* Create docker file for packaging
* Build and push docker image to the docker hub
* Run the container locally and validate

### **Report M4: Model Deployment & Orchestration**

* Setup EKS Project and Clusters
* Provide required cloud access to the collaborators
* Deployment of image using kubectl (deployment.yml)
* Validating the deployed model in the cloud

**Why EKS**

* Resource Management and Optimization: Kubernetes provides fine-grained control over resource allocation (CPU, memory), ensuring that the Flask API and ML models get the appropriate resources needed to run efficiently.
* High Availability and Reliability: EKS ensures high availability by managing multiple replicas of the Flask API and ML model across different availability zones (AZs).
* Simplified Deployment and Management: EKS simplifies the deployment of complex applications with its managed Kubernetes service. Once your Python ML model and Flask API are containerized, you can deploy them using Kubernetes resources like deployments, services, and ingress, allowing you to manage and scale the applications more easily.
* Integration with AWS Ecosystem: EKS seamlessly integrates with other AWS services like Amazon SageMaker (for advanced ML training), S3 (for storing models or data), and CloudWatch (for monitoring and logging), providing a fully managed ecosystem for your ML workflows. These integrations help reduce operational overhead and improve performance, security, and compliance.
* Security: EKS leverages AWS security features like IAM roles for access control, VPC isolation for network security, and encryption for sensitive data, ensuring that both your ML model and Flask API are secure and compliant with industry standards.

**Why Flask**

Flask is a great choice for consuming Python-based ML models due to its lightweight nature, seamless integration with Python libraries, and ability to easily expose machine learning models as RESTful APIs. It provides a simple and efficient way to deploy and scale machine learning-powered applications.

## **Github branch details:**

Repo link: <https://github.com/biju123/MLopstest>

Git clone url: <https://github.com/biju123/MLopstest.git>

# **Link for DVC Repository**

<https://github.com/biju123/MLopstest/tree/main/.dvc>

# **Link to the deployed endpoint**

<http://18.207.195.226:31233/mlops/predict_diabetes>