Foundations of DevOps

Post Assessment

Start Date: 01.05.25 End Date:07.05.25

On-Time Delivery Prediction with DevOps Integration

Objective

A logistics and e-commerce company wants to reduce delayed shipments and improve operational efficiency by using a machine learning-based system to predict whether a shipment will arrive on time or get delayed.

The company needs the solution to be:

- Deployed as a REST API using FastAPI
- Containerized for easy deployment
- Managed with Git and GitHub
- Integrated with a CI/CD pipeline using GitHub Actions

Business Need

Delivery delays have led to customer churn, increased support costs, and supply chain inefficiencies. The business seeks a solution to:

- Predict delays before dispatch
- Enable resource optimization
- Send early warnings to customers
- Improve logistics planning and customer satisfaction

Technical Requirements

Component	Requirement
Machine Learning	Binary classification model (on-time vs delayed)
Model Format	Saved as model.pkl using joblib or pickle
API Framework	FastAPI with custom request/response models
Containerization	Dockerized FastAPI application
Version Control	Git (locally) + GitHub (remote repo)
CI/CD	GitHub Actions to automate build, test, and deploy
Deployment Target	Local or DockerHub
Testing	Unit testing for model and API using pytest or unittest

Dataset

Source: https://www.kaggle.com/datasets/prachi13/customer-analytics

Features

Feature	**Description**
Warehouse_block	Source warehouse (A to F)
Mode_of_Shipment	Transport method (Flight, Ship, Road)
Customer_care_calls	Number of customer care calls
Customer_rating	Customer satisfaction (1 to 5)
Cost_of_the_Product	Product cost
Prior_purchases	Customer's prior purchase count
Product_importance	Importance level (low, medium, high)
Gender	Gender of recipient
Discount_offered	Discount offered on product
Weight_in_gms	Shipment weight

Target Feature

- Reached_on_Time_Y_N: 1 = On-Time, 0 = Delayed

Required Workflow

Model Development

- Clean, preprocess and train a binary classification model.
- Save the trained model using joblib or pickle.

FastAPI Application

- Build a FastAPI app with:
    - POST /predict endpoint to accept shipment features in JSON format
    - Prediction logic using the trained model

Dockerization

- Create a Dockerfile to:
    - Set up Python environment
    - Copy application and model files
    - Expose the FastAPI service (default: port 8000)
- Test the container locally with docker run

Git & GitHub Version Control

- Use Git for versioning
- Push to GitHub repo with structured folders

CI/CD with GitHub Actions

- Trigger: on push to main
- Steps:
    - Install dependencies
    - Run tests
    - Build Docker image
    - Push Docker image to Docker Hub (via GitHub Secrets)

Deployment

- Run the Docker container
- Interact with the API using tool curl

Submission Files

- Attach screenshots for each step and upload them in PDF format.
- Code Repository - Ensure that the repository contains all the files for deployment and upload them as a RAR file. (model.py, pkl file, app.py, requirements.txt, Dockerfile etc.,)