🏥 Hospital Inpatient Charges – End-to-End MLOps Project
📌 Project Overview

This project demonstrates an end-to-end MLOps pipeline for predicting Hospital Inpatient Charges and monitoring the model after deployment.

The focus is not only model building, but also:

Deployment

Logging

Monitoring

Data drift detection

Reproducibility

The project is built keeping real-world production constraints (free tier, dependency conflicts, data drift) in mind.

🎯 Objective

Build a machine learning model to predict Average Total Payments for hospital inpatient services.

Deploy the model as a FastAPI service.

Log predictions for monitoring.

Detect data drift using Evidently AI.

Demonstrate a complete MLOps lifecycle using free and practical tools.

🧠 Problem Statement

Hospitals generate large volumes of inpatient billing data. Over time, pricing patterns change, which can reduce model accuracy.

This project answers:

How to deploy an ML model?

How to log predictions?

How to detect when the data distribution changes?

When should the model be retrained?

🗂️ Project Structure

```
Hospital Charges for Inpatients/
│
├── app/
```

```
│   └── main.py                 # FastAPI application
│
├── data/
│   └── inpatientCharges.csv    # Training dataset
│
├── models/
│   └── inpatient_model.pkl     # Trained ML model
│
├── logs/
│   ├── predictions.csv         # Runtime logs (ignored in
git)
│   └── predictions_sample.csv  # Sample logs for GitHub
│
├── monitoring/
│   ├── drift_report.py          # Drift detection script
│   └── drift_report.html        # Generated drift report
│
├── screenshots/
│   ├── swagger_predict.png
│   ├── local_logs.png
│   ├── drift_report.png
│   └── docker_running.png
│
├── requirements.txt             # Pinned dependencies
├── Dockerfile                   # Docker configuration
├── .gitignore
└── README.md
```

⚙️ Tech Stack

Python 3.10

Pandas, NumPy, Scikit-learn

FastAPI + Uvicorn

Docker

Evidently AI (Drift Detection)

GitHub

Render (Deployment demo)

🔄 End-to-End Workflow

```
Data → Model Training → API Deployment
     → Prediction Logging
     → Drift Detection
     → Retraining Decision
```

Step-by-step:

Train ML model on historical inpatient data.

Save trained model.

Serve predictions via FastAPI.

Log prediction inputs & outputs.

Compare training vs production data.

Detect data drift.

Decide when to retrain.

🚀 How to Run the Project (Local)

1️⃣ Create & Activate Virtual Environment

```
py -3.10 -m venv venv
venv\Scripts\Activate.ps1
```

2️⃣ Install Dependencies

```
python -m pip install --upgrade pip
pip install -r requirements.txt
```

3️⃣ Run FastAPI App (Local)

```
uvicorn app.main:app --reload
```

Open Swagger UI:

http://localhost:8000/docs

4️⃣ Make Predictions

Use /predict endpoint from Swagger.

Predictions are logged into:

logs/predictions.csv

🐳 Run Using Docker (Recommended)

Build Docker Image

```
docker build -t hospital-mlops .
```

Run with Volume (for logging)

```
docker run -p 8000:8000 -v ${PWD}\logs:/app/logs hospital-
mlops
```

Swagger:

http://localhost:8000/docs

📊 Monitoring & Drift Detection

Generate Drift Report

```
python monitoring/drift_report.py
```

This generates:

monitoring/drift_report.html

What is Checked?

Feature distribution change

PSI (Population Stability Index)

Statistical tests

📈 Drift Interpretation Rules

PSI Value	Meaning	Action
< 0.10	No Drift	No action
0.10 – 0.25	Mild Drift	Monitor
≥ 0.25	Severe Drift	Retrain

🔁 Retraining Decision Logic

Retrain the model when:

Key feature shows PSI ≥ 0.25

Multiple features drift

Business metrics degrade

Periodic retraining (3–6 months)

☁️ Deployment (Render)

Dockerized app deployed on Render (Free Tier).

Due to free-tier limitations:

Persistent logging is demonstrated locally.

Cloud deployment is used for API demo only.

⚠️ Limitations (Transparent & Honest)

Render free tier does not support free persistent disks.

Logs shown via local Docker volumes.

Monitoring demonstrated offline with Evidently.

🧠 Key Learnings

Dependency pinning is critical in MLOps.

Containers are stateless by default.

Monitoring is as important as training.

Drift detection prevents silent model failure.

📃 Resume-Ready Highlights

Built and deployed an end-to-end MLOps pipeline.

Implemented prediction logging and drift detection.

Used Evidently AI for monitoring.

Handled real-world dependency conflicts.

Dockerized and deployed ML service.

📌 Conclusion

This project demonstrates real-world MLOps practices using practical and mostly free tools.
It focuses on reliability, monitoring, and reproducibility, not just model accuracy.