MediWatch: Predicting Patient Readmission with ML and MLOps

The more I learn about Machine Learning, the more I realize that *model development is only one piece of the puzzle*. Robust models are not just about getting high accuracy on a benchmark — they need to be resilient to outliers, supported by strong data and compute infrastructure, and deployed in a way that makes them usable across different environments.

Over the past year, I've been deepening my understanding of Advanced ML topics through the **Interview Kickstart Advanced Machine Learning bootcamp**. I revisited concepts like recurrent neural networks, transformers, CNNs, and even the latest LLM agents. It has been both humbling and inspiring to see how quickly the field is evolving. But what struck me most was how central **MLOps** is crucial for real-world machine learning problems.

In essence, nearly every ML challenge is an operational one:

👉 How do I manage the data versioning for model training? How do I organize new data?
👉 How do we save, version, and manage models?
👉 How do we train and re-train efficiently?
👉 How do we deploy them so they can serve predictions reliably?
👉 How do we detect when models start to drift and require updates?

Finding a "good model" is often not the hardest part. Deploying, monitoring, scaling, and maintaining it in production — that's where real impact happens.

🔗 Project code & documentation: GitHub Repository

For my **capstone project**, I chose to focus on MLOps. Together with Colin, we built **MediWatch**, an end-to-end ML system designed to predict hospital readmission risk using the well-known diabetes readmission dataset. Patient readmissions are a critical challenge for healthcare — they affect costs, strain hospital resources, and most importantly, impact patient well-being.

What we built in MediWatch:

- Developed predictive models (XGBoost was the best-performing)
- Scaled hyperparameter optimization with **Ray**
- Served models via **API**, fully containerized with **Docker & Kubernetes**
- Tracked experiments with **MLflow**
- Implemented **Drift detection** → if incoming data distribution differs from baseline, the pipeline automatically retrains and redeploys the model.

- Automated workflows using **Airflow**, to systematically retrain new models with the upcoming data.

Through this project I came to appreciate a key difference between data science and applied machine learning: data scientists often focus on extracting insights and driving business decisions, while ML engineers must ensure their models actually work *in production*, adapting them to the new data, resource constraints, and real-world latency requirements.

MediWatch became much more than a class project — for me it was a hands-on journey through the *full ML lifecycle*, from raw data to a deployed system that can adapt, scale, and improve itself over time.

I'd like to thank **Sayan** for guiding us through the MLOps concepts, **Joseph** for supporting the capstone project discussions, and **Dhiraj** for helping with exercises. This was a challenging but deeply rewarding experience — one that made me feel ready to bring these practices into real-world settings.

🚀 For me, the takeaway is clear: building models is important, but **operationalizing them is where ML creates lasting impact**.

#MachineLearning #MLOps #HealthcareAI #DataScience #ModelDeployment #MLflow #Docker #Kubernetes #RayTune #XGBoost #AI #InterviewKickstart