A/B Testing Framework for Meta’s Inference Platform

In the summer of 2022, I interned at Meta as a part of their AI Infrastructure team. My role was to build an A/B Testing Framework for Meta’s Inference Platform. By the end of the summer, I also automated the model onboarding pipeline reducing the onboarding time of new models from a couple of days to a few hours.

Project Overview

ML models are updated with new data or algorithms and deployed as they are retrained. In a highly dependent serving deployment with multi-model and multi-modal models, it is challenging to make sure a new release has no regression due to other dependent signals. One of the solutions to make sure that a new model has no regressions is to test again what is currently running. This is often called A/B testing or online experimentation.

The main goal of this project was to provide an A/B testing framework to test a newly released model for correctness before it receives full traffic in a production setting. We wanted to make sure the models change with consistent results (precision, recall, accuracy, area under precision-recall curve, etc.). This project only targeted “data” level testing, not system performance level, so it was not important to measure CPU, Memory, etc.

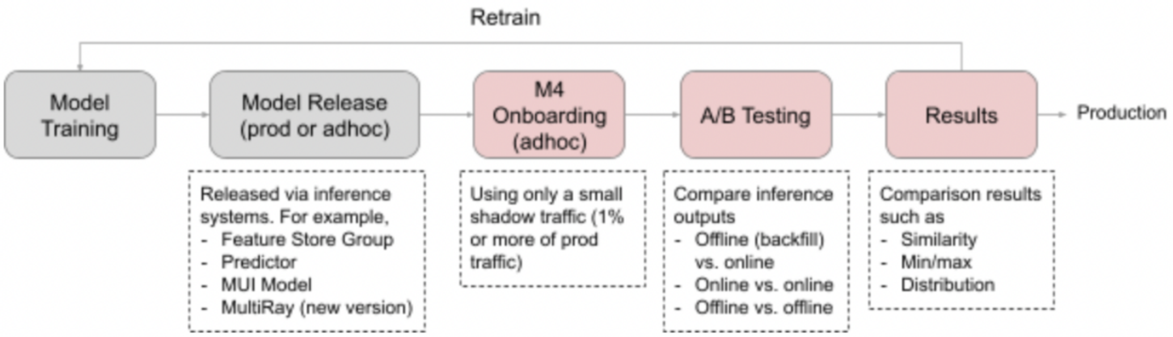
Retrain
Model Training
Model Release (prod or adhoc)
M4 Onboarding (adhoc)
A/B Testing
Results
Production
Released via inference systems. For example,
- Feature Store Group
- Predictor
- MUI Model
- MultiRay (new version)
Using only a small shadow traffic (1% or more of prod traffic)
Compare inference outputs
- Offline (backfill) vs. online
- Online vs. online
- Offline vs. offline
Comparison results such as
- Similarity
- Min/max
- Distribution

Figure. Overview of Project E2E (red boxes)

The new Inference Platform focussed on incorporating multi-model and multi-modal models was rightly named - M4. In the M4 System, the models could be organized as nodes in a DAG. In this DAG Framework, data would come to a node, would be processed, and be pushed to downstream nodes. The structure of the DAG could be defined in configuration files written in python. Whereas, the individual processing logic was written in C++. While working on this system I interacted with a lot of other platforms like Meta’s Feature Store which is essentially an interface to allow users to compute features from data stored in the backend and Media Understanding Engine which is the infrastructure used by the team that works with video and image data.

Right now, every team in Meta has its own inference framework/platform which is tailored to its needs. The goal of the M4 project is to be a one-stop shop for unified serving and unified authoring.

The goal of unified serving for this service is to be able to be used for not only multiple models but multiple types of models - serving video, image, text, etc. Another goal is to allow users to specify the DAG structure and the properties of the DAG in an extensive but simple manner.

Internship Work

Since M4 aimed to be a unified serving and unified authoring system, my first goal was to make the onboarding experience as smooth as possible. At the time of this work, we were migrating models from the old infrastructure to the new M4 System. This was done manually by the M4 team, but the goal was to make the process so easy that the model owners can do it themselves. To understand how model migration works, I lead the migration of an important model to M4. During this time, I understood some of the manual processes which can be automated. So in the first 4 weeks, after ramping up, I created two scripts that would automate some of the migration work. These scripts were used for this particular migration and will be used in the future for more migrations.

After understanding and improving the onboarding process, I worked on the A/B Testing framework for the M4 System. After a lot of design discussions with my team, we settled on the approach where we introduce a new type of node in the DAG Framework called "Signal Comparison Node". This node would take data from their parent nodes, perform the comparison(A/B Testing) in the backend and push the data to a database. In the next few weeks, I implemented this approach and demonstrated how one could use A/B testing on the new model that I helped migrate. I also developed a dashboard where users could see a graph of the A/B Testing comparison results for richer feedback.

My third and final task was going back to the onboarding experience and improving it further. Based on the information available from different sources, I wanted to allow users to generate the whole DAG using just one command line invocation. In the last couple of weeks of my internship, I worked on implementing this. This was a stretch goal for me and I was able to complete 80% of this task.

Toward the end, I worked on writing documentation for my code and worked on handing it over to the new engineers on the team who will take over this work.

Learnings

As for my learnings I learned a lot during my internship

- I learned how Meta deploys Systems for ML in their production systems.
- I learned about the workflow of ML Engineers and how their work integrates with that of the Infra team.
- I learned about the distributed system that Meta uses for serving their models. Not just one model, but multiple types of models in the same framework.
- I learned how to talk to different stakeholders and learn about their requirements. I would then use this information to design the system I was working on.