014487800, 014997604

Use case is that we have a hobby recommender, and the ground truth is gained via user updates on their hobbies on this platform. This update frequency is mentioned to be highly random. This recommender was trained once with a hefty sample or training data. So how do we know if it is still working?

One way to monitor if our model still predicts okey would be to have a null model as reference point. This null model could be just simply randomly predicting hobbies and then comparing these results with our actual model: if our model performs as good as the random suggestions maybe it would be a cue to retrain our actual model again. This requires tracking and logging the ground truth, that is whether user updated their hobbies after recommendations. In addition to null model, we could have some sort of shadow deployment of a candidate model that keeps updating when certain amount of new data (updated hobbies) is available. This candidate model would then replace our actual model when pre-defined threshold would be achieved.

For this we would need to have experiment tracking and logging some model performance metrics, especially above-mentioned ground truth. Even thou the frequency could be low it would still be well worth tracking since this is the data we can evaluate our model error rates and such.

To extrapolate more on this shadow deployment, it would be useful to use slice-based evaluation in background, training the model with all recent data that we have been able to gather with the ground truth. This shadow model would then be evaluated (automated) against the original model with some accuracy-based metrics.

One other way to assess if our model needs to be re-trained would be to monitor different shifts in our data distributions and model performance. This kind of a pipeline could include two-sample tests and statistical methods to test whether the distribution has shifted. Standard drift detection methods include DDM, which monitors the error rate of the predictive model over time, and ADWIN, which monitors the response distribution within a dynamic sized window. Of course, this would be automated so that our model gets re-trained automatically when certain limits or thresholds are met, or data scientists would be alerted if some drifts has been detected.

The ground truth is determined to be whether the user updates the recommended hobby to their profile hobby list or not, that is the probability that recommended hobby is adopted as a new hobby by the user and that the user updates it to the system. Alternatively, the ground truth could be determined to be the user feedback of the suggestions, such as a upvote/downvote feedback, or even rating the suggestions from 1 to 5 starts for example.

To increase the likelihood of getting new data with ground truth we could incentivize this somehow, for example discount coupon to online hobby sites or so per ten ratings or hobby list updates.

I would say that this pipeline could be executed with Kubernetes (containerization, serving), MLFlow (experiment tracking, model registry), Evidently (data drift) and Prometheus (system monitoring); which with all we have been experimenting with throughout this course.