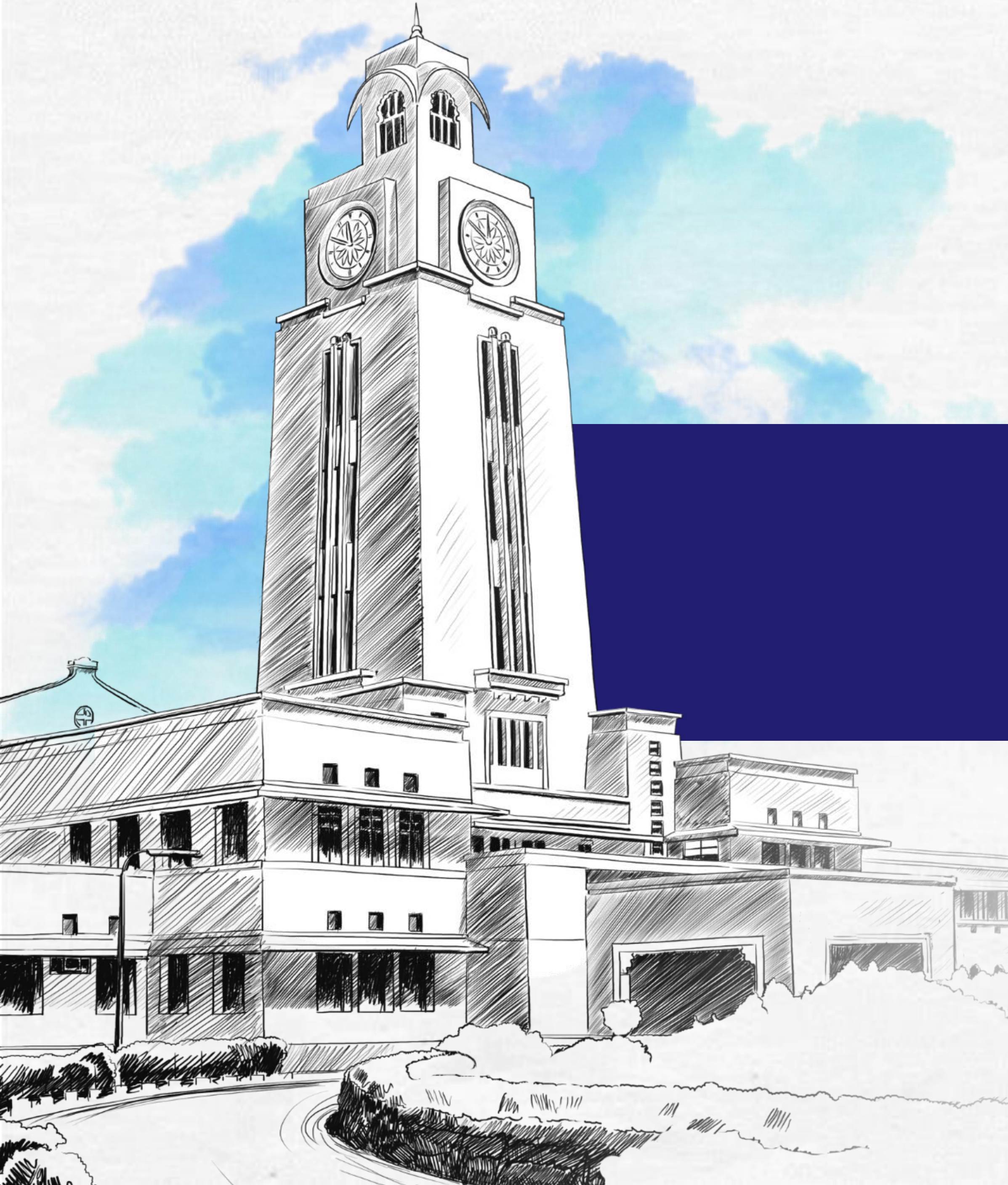


Monitoring and Feedback Loop

Pravin Y Pawar

Adapted from “Introducing MLOps”
By Treveil et al

Model monitoring

Model monitoring is a crucial step in the ML model life cycle and a critical piece of MLOps

- When a machine learning model is deployed in production
  - it can start degrading in quality fast—and without warning—until it's too late

- Machine learning models need to be monitored at two levels:

- At the resource level, including ensuring the model is running correctly in the production environment.
  - Key questions include:
    - Is the system alive?
    - Are the CPU, RAM, network usage, and disk space as expected?
    - Are requests being processed at the expected rate?

- At the performance level, meaning monitoring the pertinence of the model over time
  - Key questions include:
    - Is the model still an accurate representation of the pattern of new incoming data?
    - Is it performing as well as it did during the design phase?

Model performance monitoring

- The first level is a traditional DevOps topic

- The latter is more complicated. Why?
  - Because how well a model performs is a reflection of the data used to train it
    - in particular, how representative that training data is of the live request data

- As the world is constantly changing,
  - a static model cannot catch up with new patterns that are emerging and evolving without a constant source of new data

- Model performance monitoring attempts to track this degradation,
  - At an appropriate time, it will also trigger the retraining of the model with more representative data

Model Retraining

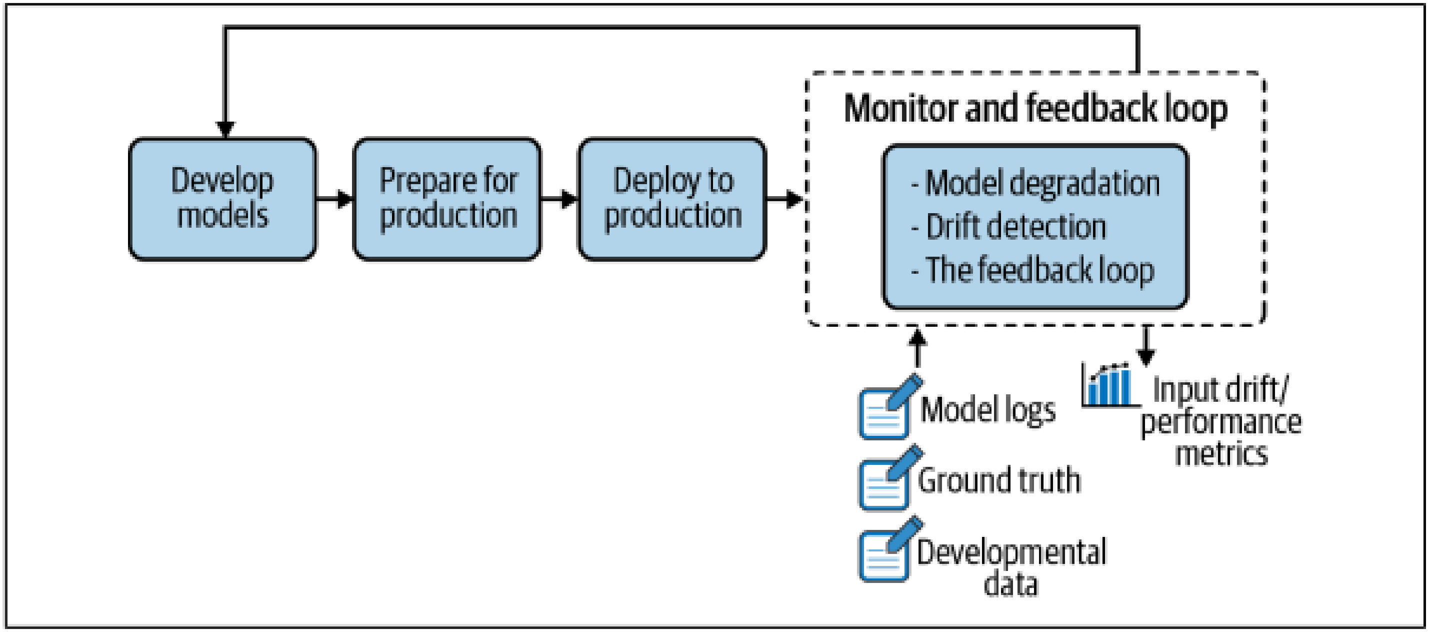
- Critical for organizations to have a clear idea of deployed models' drift and accuracy
  - by setting up a process that allows for easy monitoring and notifications

- Ideal scenario would be a pipeline that automatically triggers checks for degradation of model performance
  - goal of notifications is not necessarily to kick off an automated process of retraining, validation, and deployment
  - but to alert the data scientist of the change; to diagnose the issue and evaluate the next course of action

- Critical that as part of MLOps and the ML model life cycle,
  - data scientists and their managers and the organization as a whole understand model degradation

- Every deployed model should come with monitoring metrics and corresponding warning thresholds
  - to detect meaningful business performance drops as quickly as possible

Monitoring and feedback loop



Monitoring and feedback loop highlighted in the larger context of the ML project life cycle

Model Degradation

- Once a machine learning model is trained and deployed in production
- Two approaches to monitor its performance degradation:
  - ground truth evaluation
  - input drift detection

Ground Truth Evaluation

Defined

- Ground truth retraining requires waiting for the label event
  - In a fraud detection model, the ground truth would be whether or not a specific transaction was actually fraudulent
  - For a recommendation engine, it would be whether or not the customer clicked on—or ultimately bought—one of the recommended products

- With the new ground truth collected,
  - next step is to compute the performance of model based on ground truth
  - compare it with registered metrics in the training phase

- When the difference surpasses a threshold, the model can be deemed as outdated, and it should be retrained

- The metrics to be monitored can be of two varieties:
  - Statistical metrics like accuracy, ROC AUC, log loss, etc.
    - domain agnostic
    - drawback is that the drop may be statistically significant without having any noticeable impact
    - cost of retraining and risk associated with a redeployment may be higher than expected benefits

  - Business metrics, like cost-benefit assessment such as the credit scoring
    - far more interesting because they ordinarily have a monetary value
    - enabling subject matter experts to better handle the cost-benefit trade-off of the retraining decision

Ground Truth Evaluation(2)

Challenges

- Ground truth is not always immediately, or even imminently, available
  - For some types of models, teams need to wait months (or longer) for ground truth labels to be available,
  - which can mean significant economic loss if the model is degrading quickly
  - Deploying a model for which the drift is faster than the lag is risky

- Ground truth and prediction are decoupled
  - To compute the performance of the deployed model on new data, it's necessary to be able to match ground truth with the corresponding observation
  - In many production environments, this is a challenging task because these two pieces of information are generated and stored in different systems and at different timestamps

- Ground truth is only partially available
  - In some situations, it is extremely expensive to retrieve the ground truth for all the observations,
  - which means choosing which samples to label and thus inadvertently introducing bias into the system

Input Drift

- Mathematically speaking, the samples of each dataset cannot be assumed to be drawn from the same distribution
  - i.e., they are not “identically distributed”

- Another mathematical property is necessary to ensure that ML algorithms perform as expected: independence
  - property is broken if samples are duplicated in the dataset or if it is possible to forecast the “next” sample given the previous one

- If train the algorithm on the first dataset and then deploy it on the second one
  - The resulting distribution shift is called a drift

- In wine quality prediction exercise,
- Called a feature drift
  - if the alcohol level is one of the features used by the ML model
  - or if the alcohol level is correlated with other features used by the model
- Called as a concept drift if it is not

Drift Detection in Practice

- To be able to react in a timely manner, model behavior should be monitored solely based on the feature values of the incoming data,
  - without waiting for the ground truth to be available

- The logic is that if the data distribution (e.g., mean, sd, correlations between features) diverges
  - between the training and testing phases on one side and the development phase on the other,
  - a strong signal that the model's performance won't be the same!

- Not the perfect mitigation measure, as retraining on the drifted dataset will not be an option,
  - but it can be part of mitigation measures (e.g., reverting to a simpler model, reweighting)

Causes of Data Drift

Two frequent root causes of data drift

- Sample selection bias
  - the training sample is not representative of the population
  - often stems from the data collection pipeline itself
  - For instance, building a model to assess the effectiveness of a discount program will be biased
    - if the best discounts are proposed for the best clients
- Non-stationary environment
  - where training data collected from the source population does not represent the target population
  - Often happens for time dependent tasks — such as forecasting with strong seasonality effects,
    - where learning a model over a given month won't generalize to another month

Input Drift Detection Techniques

Choice depends on the expected level of interpretability

- Two approaches
  - Univariate statistical tests
  - Domain classifier

- Should prefer univariate statistical tests
  - Organizations that need proven and explainable methods

- Domain classifier approach may be a good option
  - if complex drift involving several features simultaneously is expected
  - if the data scientists want to reuse what they already know and assuming the organization doesn’t dread the black box effect

Univariate statistical tests

- Requires applying a statistical test on data from the source distribution and the target distribution for each feature
  - A warning will be raised when the results of those tests are significant

- Basic approaches rely on two tests:
  - For continuous features, the Kolmogorov-Smirnov test is a nonparametric hypothesis test that is used to check whether two
  samples come from the same distribution
  - For categorical features, the Chi-squared test is a practical choice that checks whether the observed frequencies
  for a categorical feature in the target datamatch the expected frequencies seen from the source data

- The main advantage of p-values is that they help detect drift as quickly as possible
- The main drawback is that they do not quantify the level of the effect

- If development datasets are very large, it is necessary to complement p-values with business-significant metrics
  - For example, on a sufficiently large dataset, the average age may have significantly drifted from a statistical perspective,
  - but if the drift is only a few months, this is probably an insignificant value for many business use cases

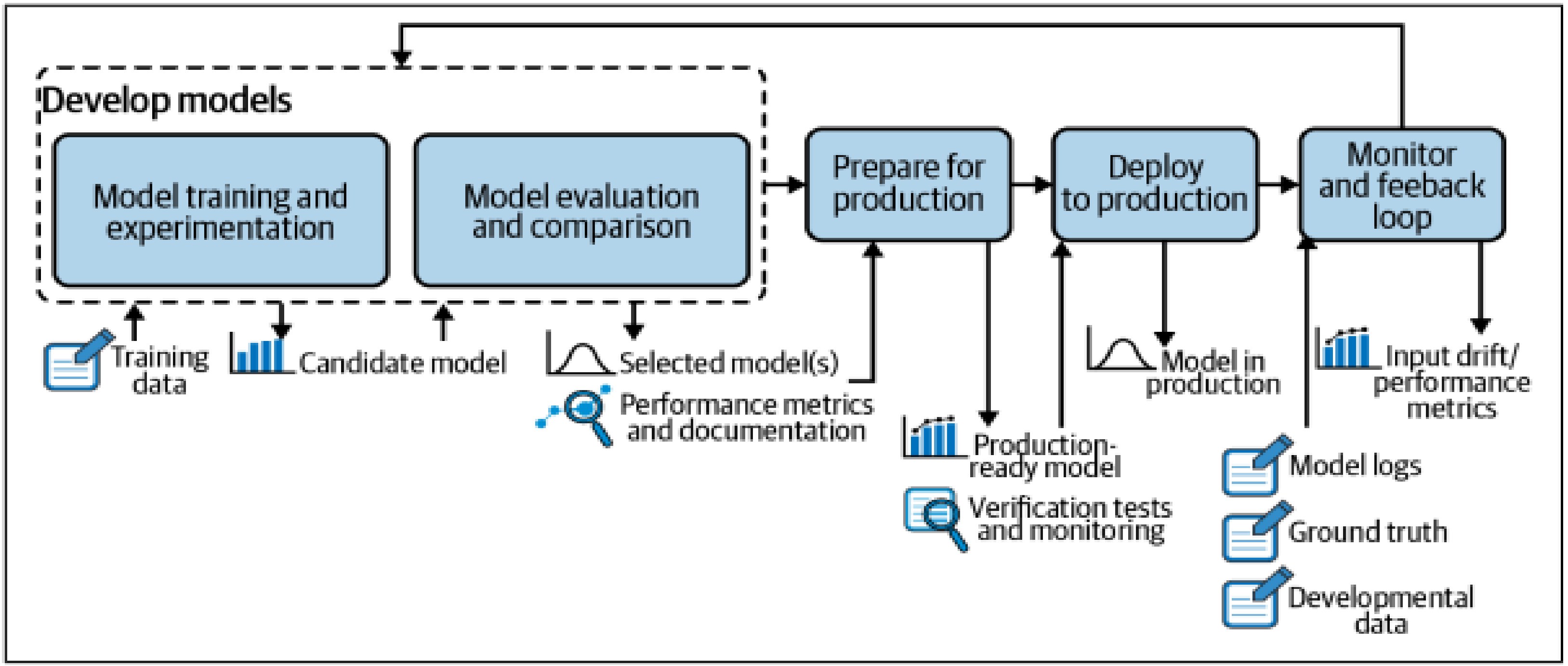
Domain classifier

- Data scientists train a model that tries to discriminate between the original dataset (input features and, optionally, predicted target) and the development dataset
  - stack the two datasets and train a classifier that aims at predicting the data’s origin

- The performance of the model (its accuracy, for example) can then be considered as a metric for the drift level
  - If this model is successful in its task, and thus has a high drift score,
  - implies that the data used at training time and the new data can be distinguished
  - fair to say that the new data has drifted

- To gain more insights, in particular to identify the features that are responsible for the drift
  - one can use the feature importance of the trained model

Continuous delivery for end-to-end machine learning process



Continuous delivery for end-to-end machine learning process

The Feedback Loop

- All effective machine learning projects implement a form of data feedback loop
  - information from the production environment flows back to the model prototyping environment for further improvement

- Continuous delivery for end-to-end machine learning process
  - Data collected in the monitoring and feedback loop is sent to the model development phase
  - From there, the system analyzes whether the model is working as expected
  - If it is, no action is required
  - If the model's performance is degrading, an update will be triggered, either automatically or manually by the data scientist

- In practice, usually means either retraining the model with new labeled data or developing a new model with additional features

The Feedback Loop(2)

- Goal of retraining is to be able to capture the emerging patterns and make sure that the business is not negatively impacted

- This infrastructure is comprised of three main components, which are critical to robust MLOps capabilities:
  - A logging system that collects data from several production servers
  - A model evaluation store that does versioning and evaluation between different model versions
  - An online system that does model comparison on production environments,
    - either with the shadow scoring (champion/challenger) setup or with A/B testing

Logging

- Monitoring a live system means collecting and aggregating data about its states

- Nowadays, as production infrastructures are getting more and more complex,
  - with several models deployed simultaneously across several servers,
  - an effective logging system is more important than ever!

- Data from these environments needs to be centralized to be analyzed and monitored,
  - either automatically or manually
  - will enable continuous improvement of the ML system

- An event log of a machine learning system is a record with a timestamp and information such as
  - Model metadata - Identification of the model and the version
  - Model inputs - Feature values of new observations
  - Model outputs - Predictions made by the model that
  - System action - an action taken based on model prediction
  - Model explanation - an explanation of prediction (i.e., which features have the most influence on the prediction

Model Evaluation

- If model performance is degrading, after review, data scientists decide to improve the model by retraining it,
  - With several trained candidate models, the next step is to compare them with the deployed model
  - means evaluating all the models (the candidates as well as the deployed model) on the same dataset

- If one of the candidate models outperforms the deployed model, there are two ways to proceed:
  - either update the model on the production environment
  - or move to an online evaluation via a champion/challenger or A/B testing setup

- In a nutshell, this is the notion of model store
- A structure that allows data scientists to:
  - Compare multiple, newly trained model versions against existing deployed versions
  - Compare completely new models against versions of other models on labeled data
  - Track model performance over time

Model evaluation store

- Formally, the model evaluation store serves as a structure that centralizes the data related to model life cycle to allow comparisons

- Two main tasks of a model evaluation store are:
  - Versioning the evolution of a logical model through time
    - Each logged version of the logical model must come with all the essential information concerning its training phase, including:
      - The list of features used
      - The preprocessing techniques that are applied to each feature
      - The algorithm used, along with the chosen hyperparameters
      - The training dataset
      - The test dataset used to evaluate the trained model (this is necessary for the version comparison phase)
      - Evaluation metrics
  - Comparing the performance between different versions of a logical model

Online Evaluation

- Online evaluation of models in production is critical from a business perspective,
    - but can be challenging from a technical perspective

- Two main modes of online evaluation:
    - Champion/challenger (otherwise known as shadow testing)
        - the candidate model shadows the deployed model and scores the same live requests
    - A/B testing
        - the candidate model scores a portion of the live requests and the deployed model scores the others

- Both cases require ground truth
    - evaluation will necessarily take longer than the lag between prediction and ground truth obtention

- Whenever shadow testing is possible, it should be used over A/B testing
    - because it is far simpler to understand and set up, and it detects differences more quickly

Thank You!

In our next session: