

Production: Data Distribution Shifts and Monitoring

```
$ echo "Data Science Institute"
```

Agenda

7.1. Monitoring

- ML System Failures
- Data Distribution Shifts
- Monitoring and Observability

7.2 Continual Learning and Test in Production

- Testing data distribution shifts.

Slides, Notebooks, and Code

- These notes are based on Chapter 6 of *Designing Machine Learning Systems*, by [Chip Huyen](#).

Notebooks

- `./01_materials/labs/production_7_distribution_shifts.ipynb`

Our Reference Architecture

The Flock Reference Architecture

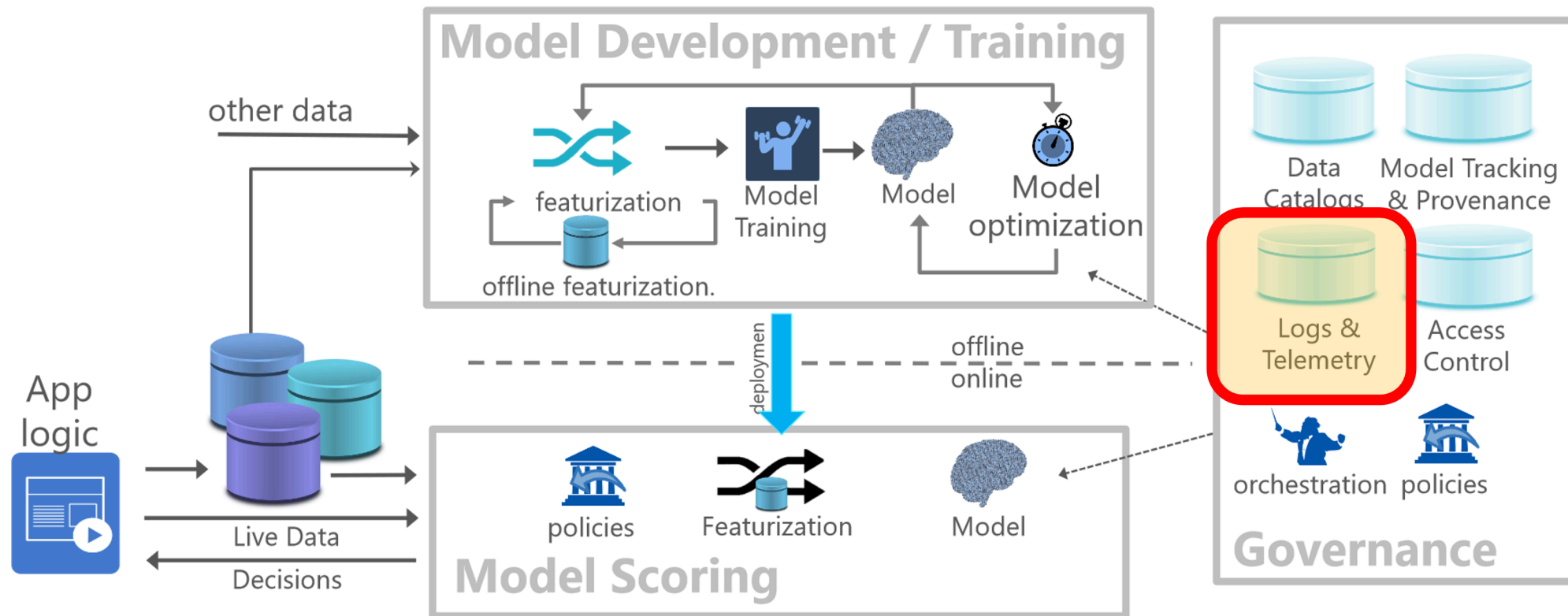


Figure 1: Flock reference architecture for a canonical data science lifecycle.

ML System Failures

What is an ML System Failure?

- A failure happens when one or more expectations of the system is not met:
 - Traditional software expectations: the system executes its logic within the expected metrics, such as latency and throughput.
 - ML performance: performance metrics are met, explanations are given, trust in the system (can be achieved by communicating uncertainty), etc.

What is an ML System Failure?

- Operational expectations can be easier to detect than ML performance expectations.
- Understanding why ML systems fail can help monitor ML performance.

Software System Failures

- Dependency failure
 - A package or codebase that the system depends on breaks, which leads to system failure.
 - Common when the dependency is maintained by a third party.
 - Can also happen when our model is a dependency of a downstream consumer.
- Deployment failure
 - The root cause is deployment errors: deploy old binaries, permissions are not correctly granted, etc.
 - Coding errors and integration errors (interface changes).

Software System Failures

- Hardware failures
 - Hardware use to deploy the model fails.
- Downtime or crashing
 - Connectivity, security, and other issues may give rise to unreachable servers (AWS, Azure, GCP, etc.)
 - Distributed systems are complex systems, and the risk of failure increases with complexity.

ML-Specific Failures

Production data is different from training data

- A key assumption is that training and unseen data come from the same distribution.
- When we say that a model *learns* from data, we are saying that the model learns the distribution of the training data to use this information on unseen data.
- When predictions on unseen data are satisfactory, we say the model "generalizes to unseen data".
- The test data used in the model development phase and the cross-validation are *estimates* of the error in unseen (production) data.
- Reasons for difference:
 - Data collection, encoding, and instrumentation.
 - The world changes, and data distributions are not stationary.

ML-Specific Failures

Edge cases

- An ML model performs well in most cases but fails in a small minority of cases, generally with catastrophic consequences.
- Data distribution may have shifted if the number of edge cases increases.
- Key concern for safety-critical applications: autonomous vehicles, health systems, risk monitoring, etc.

ML-Specific Failures

Degenerate feedback loops

- The model's predictions influence the feedback, which in turn influences the next iteration of the model:
 - System outputs are used to generate the next set of inputs.
 - In user-facing applications, this can drive the options or interactions that a user is offered.
 - User interactions with the system are the training data.

Data Distribution Shifts

Types of Data Distribution Shifts

Three types of shifts:

- Concept drift
- Covariate shift
- Label shift

Before we begin:

- Assume that we are looking to predict Y given data X .
- To do so, we estimate $P(Y|X)$.
- Our data, shows a distribution $P(X, Y)$ and we know that:

$$P(X, Y) = P(Y|X)P(X)P(X, Y) = P(X|Y)P(Y)$$

Types of Distribution Shifts

- Covariate shift:
 - $P(X)$ changes.
 - $P(Y|X)$ does not change.
- Label shift:
 - $P(Y)$ changes.
 - $P(X|Y)$ does not change.
- Concept drift:
 - $P(Y|X)$ changes.
 - $P(X)$ does not change.

Types of Distribution Shifts

$$P(X, Y) = P(Y|X)P(X)P(X, Y) = P(X|Y)P(Y)$$

- $P(X, Y)$ joint distribution.
- $P(Y|X)$ conditional probability of output Y given input X.
- $P(X)$ probability density of input.
- $P(Y)$ probability density of output.

Covariate Shift

- Covariate shift:
 - $P(X)$ changes.
 - $P(X|Y)$ does not change.
- Widely studied distribution shifts.
- Covariate is an independent variable that can influence the outcome of a statistical trial but it is not of direct interest.
- Example: while predicting house prices as a function of location, a covariate is square footage.

Covariate Shift

Causes:

- Sampling methods: example, oversampling of cancer patients over 40.
- Training data is artificially altered: applied SMOTE and distribution changed.
- Active learning: instead of randomly sampling, use samples most helpful to that model according to some heuristic.
- Major changes in the production environment or application: changes in marketing, for example, induce more clients from a certain demographic not previously represented in training data.

Label Shift

- Label shift:
 - $P(Y)$ changes.
 - $P(Y|X)$ does not change.
- Also known as *prior shift*, *prior probability shift*, or *target shift*.
- The output distribution changes, but for a given output, the input distribution stays the same.
- When a covariate shift happens, it could be followed by a label shift.
- Methods for detecting covariate and label shifts are similar.

Concept Drift

- Concept drift:
 - $P(Y|X)$ changes.
 - $P(X)$ does not change.
- Also known as *posterior drift*.
- Input distribution remains the same, but the conditional distribution of the output given an input changes.
- "Same input, different output."
- Can be cyclic or seasonal.

Detecting Data Distribution Shifts

Exploratory Data Analysis

- Compare different quantiles of data distributions and compare: 5th, 25th, 50th, 75th, and 95th.
- Comparing mean, median, and standard deviation only may give partial results:
 - Noticing differences may be indicative of a distribution shift.
 - **Not** noticing differences could hide distribution shifts.

Detecting Data Distribution Shifts

Statistical methods

- A more robust approach is to use two-sample hypothesis tests.
- These tests help us determine if the difference between distributions is statistically significant: if it is, then the probability that the difference is due to random fluctuations is low.
- If a difference is detected, it does not necessarily mean it is important. However, if the difference is noticeable in a small sample, it generally indicates that it is important.
- Many methods for univariate data. For example, Kolmogorov-Smirnov test.
- Other methods for multivariate data: Least Squares Density Difference or Maximum Mean Discrepancy.
- In general, these methods are better for low dimensional data; it may be

Drift Detection Methods

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Kolmogorov-Smirnov	✓	✓		✓	✓		✓
Cramér-von Mises	✓	✓				✓	✓
Fisher's Exact Test	✓				✓	✓	✓
Maximum Mean Discrepancy (MMD)	✓	✓		✓	✓	✓	
Learned Kernel MMD	✓	✓		✓	✓		
Context-aware MMD	✓	✓	✓	✓	✓		
Least-Squares Density Difference	✓	✓		✓	✓	✓	
Chi-Squared	✓				✓		✓
Mixed-type tabular data	✓				✓		✓
Classifier	✓	✓	✓	✓	✓		
Spot-the-diff	✓	✓	✓	✓	✓		✓
Classifier Uncertainty	✓	✓	✓	✓	✓		
Regressor Uncertainty	✓	✓	✓	✓	✓		

Monitoring and Observability

Monitoring and Observability

Monitoring

- Tracking, measuring, and logging different *metrics* that can help determine when something goes wrong.
- Classes of metrics to monitor:
 - Operational: convey the health of the system. Operational metrics are related to network, machine, and application. Ex.: latency, throughput, prediction requests per unit of time, percentage of successful predictions, CPU/GPU utilization, memory use, etc.
 - ML Specific Metrics: model performance, predictions, features, and raw inputs.

Monitoring and Observability

Observability

- Setting up a system in a way that affords us visibility into its inner workings to help us investigate it to solve bugs and produce enhancements.
- Logs and reporting.
- Instrumentation and telemetry.

Monitoring ML Systems

- Monitoring model performance:
 - Prediction correctness is only part of the story.
 - Collect performance in terms of usability and trust (preferences).
 - Collect inferred metrics (clicks, accepted recommendations, etc.)
- Monitoring predictions
 - Monitor distribution shifts.
 - Slice analysis, backtesting, etc.

Monitoring and Observability

- Monitoring features
 - Monitor input features and transformed features.
 - Easier to validate than raw data because a defined schema exists for features.
 - Common validation tests:
 - Min, max, median, and other quantile values.
 - Values satisfy a certain regular expression.
 - Values belong to a predefined set.
 - Values of a feature are always positive, less than one, greater than another feature's value, etc.
- Monitoring raw data
 - Generally, a responsibility of the data engineering team or data governance.
 - Automated pipelines and data quality verification

References

References

- Agrawal, A. et al. "Cloudy with a high chance of DBMS: A 10-year prediction for Enterprise-Grade ML." arXiv preprint arXiv:1909.00084 (2019).
- Huyen, Chip. "Designing machine learning systems." O'Reilly Media, Inc.(2021).