iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

MANAGING OBSERVABILITY FOR NON-DETERMINISTIC WORKLOADS IN AI AND ML SYSTEMS

Vignesh Kumar Subramanian
Senior Software Engineer
MarketAxess
20 Beacon Way, Jersey City, New Jersey, USA
Kumarsubramanian.v@northeastern.edu

ABSTRACT
Today, Artificial Intelligence (AI) and Machine Learning (ML) systems are moving across all industries where critical decisions are driven, and their non-deterministic nature is challenging to effectively monitor and hold accountable. In this paper, we examine the limitations of typical monitoring tools to capture the intricate, dynamic behavior of AI/ML workloads and advocate toward augmenting to a more fortified and contextual observation structure. The paper examines the technical hurdles and real-life problems that undermine observability, as well as the standard technical components, including data quality validation, model performance tracking, explainability, service tracing, and real-time drift detection. The study then demonstrates how leading organizations incorporate observability practices into the ML lifecycle through an analysis of real-world implementations at Uber, Airbnb, and Netflix. Additionally, we review the existing ecosystem of tools, including infrastructure monitors as well as ML personalized ones, and offer best practices such as metadata logging, automated alerts, privacy-aware traces, and explainability integration. The work concludes on future directions in observability, such as causal observability and federated tracing, but with a clear punchline: observability is an underlying cornerstone to building trustworthy, reliable, and ethical AI systems.

Keywords:
Non-Deterministic AI/ML Systems, Model Drift Detection, Feature Drift, Explainable AI (XAI), Data Quality Monitoring, Observability in ML Systems, Real-Time Performance Monitoring, Model Explainability Tools (SHAP, LIME), Causal and Federated Observability

INTRODUCTION

Artificial intelligence and machine learning (AI and ML) technologies have revolutionized almost all developed sectors, including healthcare, finance, transportation, e-commerce, etc. The most complex models to date of any kind are being embedded in the operational infrastructure of organizations, and there is an increasing ability to understand, debug, and even trust these systems. Until recent technological advances, system health and reliability were maintained using monitoring practices that had highly deterministic system behaviors that produced predictable outputs under consistent inputs. However, AI and ML systems are less predictable. Instead, they typically display nondeterministic behaviors due to stochastic processes in training, how probabilistic inference is performed, failed and adapting datasets, and the use of adaptive algorithms. Traditional observability practices are insufficient to build and maintain the reliability, interpretability, and robustness of AI/ML workloads due to these characteristics [1].

While monitoring merely observes the system from the outside, observability consists of the ability to infer the internal state of the system from its external outputs. For nondeterministic workloads, observability is required to not just include metrics like CPU Usage and Memory Allocation but also include the model accuracy, data drift, feature importance, and real-time decision logic. This work investigates the evolving realm of observability in AI/ML systems in light of the issues of non-determinism and how such problems must be handled. The discussion touches on how real-world use cases and best practices make use of current tooling, through the analysis of use cases and today's

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

tooling, to uncover and provide a thorough understanding of how organizations can increase trust and accountability in AI/ML systems via increased observability.

Understanding Non-Deterministic AI/ML Systems

One major challenge in observability for AI and machine learning (ML) systems lies in the fact that observability for AI and machine learning (ML) systems is inherently nondeterministic. AI/ML systems are different in that their 'inputs' do not always lead to consistent outputs as you might expect in traditional deterministic systems. Various factors give rise to this nondeterminism. In training, stochastic processes, like Stochastic Gradient Descent (SGD), add randomness for better generalization. Dropout layers, sampling methods, and randomized decision paths, in particular, may be used in inference processes of ensemble or Bayesian models so that the resulting outputs vary. Continuous learning systems, on the other hand, in the production setting may retrain the models from real-time data, but there may be no version control or clear traceability. Also, deployment in a distributed environment further introduces uncertainty, as asynchronous updates, race conditions, and even infrastructure inconsistencies can change the behavior [2].

Although these traits help to be flexible and for learning new behaviors, they make it harder to trace or understand how the model behaves. One incorrect output could not have a consistent pattern to it and may just be due to a shift in the data, poor quality of input data, or previously unseen cases. For instance, a reduction in model accuracy may be the result of a silent data distribution drift rather than a technical fault. The complexity makes traditional observability tools (e.g., CPU usage or error log) insufficient. Instead, effective observability in such environments requires knowledge of system behaviors in an ambient, context-aware framework that includes data quality, model logic, external dependencies, and temporal dynamics. Only then can teams fully control AI/ML system performance in ever-changing, real-world conditions and monitor, diagnose, and improve performance [3].

Limitations of Traditional Monitoring for AI/ML Workloads

For infrastructure observability, traditional monitoring tools, including Nagios, Prometheus, and New Relic, have taken their place for a long time. These metrics are low-level and are used by them to perform tracking of CPU utilization, memory consumption, disk I/O, network traffic, and system uptime. Such indicators are indispensable for keeping servers, containers, and applications operationally healthy. But these tools are lacking when it comes to using them for AI and ML workloads to gain insights to determine model performance, fairness, or decision quality.

We operate at a higher abstraction layer, and success is not only about uptime and latency (although these are important in their own right) but about how accurately and reliably the models that machine learning systems leverage perform for the actual tasks they have been charged with performing. While a model could seem fully functional, it might still be degrading in computing and performing badly. Model drift, data distribution shift, or the emergence of biases can lead to this. Because these problems are mostly invisible to regular monitoring setups (i.e., setups without semantic understanding necessary to judge guess correctness, data quality, or feature behavior), they are encountered often.

Also, AI/ML pipelines contain multiple asynchronous components of data ingestion, preprocessing, training, validation, inference, and retraining continuously. Also, these stages cover several tools and systems that often involve external or dynamic data sources. Such a thing as a mislabeled training sample or a broken feature transformation will not generate any alerts in standard logs or metrics, but can very much affect the model outcome. In the production environment, such a disconnect can be a big risk, as unchecked problems can do irreparable damage to products due to flawed decision-making [4].

ML systems need domain-specific observational methods to go beyond system metrics, which guarantees effective monitoring. Observability requires model-specific indicators, including feature drift assessments along with label drift indicators, as well as prediction reliability indices and class distribution monitoring and explanations to discover AI pipeline issues promptly.

Core Components of Observability in AI/ML Systems

A full-scale method combining various levels of observation must replace standard infrastructure measurements to handle observability in non-deterministic AI/ML systems. System performance tracking and diagnosis and enhancement require observability, which consolidates information from data quality and model behavior along with system infrastructure and user interactions. The total system health and adaptive capacity emerge from collaborated components, which allow tracking emerging problems before they occur [5].

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

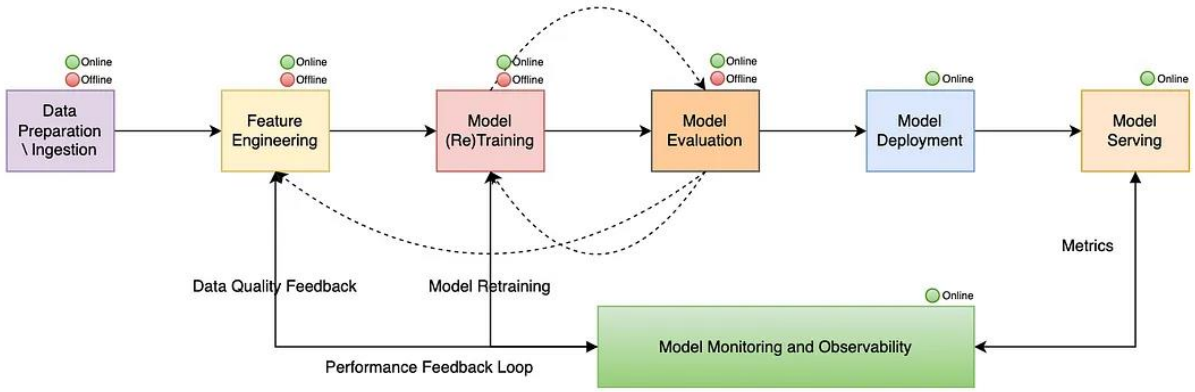

Image 1: AI/ML Observability Architecture Diagram [6]

Data Quality Monitoring

Data quality monitoring represents the fundamental starting point for observability within AI/ML systems' infrastructure. Machine learning models heavily depend on their processing data; thus, maintaining quality data integrity remains essential. System checks for real-time data validation should verify the completion of values and eliminate unexpected null entries, as well as look out for outliers and violations of schemas while detecting inconsistent encoding patterns and sampling inconsistencies. Observable systems require live statistical property checks for mean values and variance measurements alongside correlation analysis for spotting deviations in feature data distributions and data drift occurrences. Machine-generated alert systems identify upstream data problems prior to their impact on model predictions by detecting these property deviations [7].

Feature Drift and Model Performance Tracking

The observability system requires detecting feature drift and continuously monitoring performance as a crucial element. When systems rely on user-generated or external data within dynamic environments, the features that constitute their inputs tend to transform with the passage of time. Tools need to implement Population Stability Index (PSI), Kullback–Leibler divergence (KLD), and Jensen–Shannon distance (JSD) performance metrics to evaluate feature drift identically. The monitoring process for real-time performance analysis needs to incorporate precision and recall along with the F1-score, area under the curve (AUC), and calibration error data. The absence or delay of ground truth labels can be monitored through surrogate indicators that include prediction entropy along with confidence scores and model output variance, which detect potential performance degradation [8].

Model Explainability and Debugging

Model complexity demands a clear decision rationale for both accountability needs and trust establishment in the system. Model decision deconstruction becomes possible through SHAP (Shapley Additive explanations) combined with LIME (Local Interpretable Model-agnostic Explanations), which reveals how components of input data affect resulting outputs. Observability at this level helps data scientists locate hidden biases as well as discover incorrect logic while it enables effective communication with stakeholders.

Cross-Service Observability and Lineage Tracking

The contemporary ML systems integrate models within bigger distributed systems. A tracking system measures communication across various system elements, including data pipelines, along with feature stores, model servers, and API infrastructure. OpenTelemetry provides distributed tracing as a tool, while MLflow and DVC enable data and model versioning to provide traceability and reproductive capabilities, as well as efficient negative scenario rollback [9].

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

Table 1: Metrics for Monitoring AI/ML Systems
It summarizes key metric types used to monitor AI/ML systems, focusing on performance, data integrity, confidence, and operational efficiency.

Metric Type	Description	Purpose	Example Metric
Model Performance Metrics	Metrics that track how well the model is performing over time.	Helps detect model drift and assess model accuracy.	Precision, Recall, F1-Score, AUC-ROC
Feature Drift Metrics	Metrics that measure shifts in input data distributions.	Identifies when input features deviate from the training data.	Population Stability Index (PSI), Jensen–Shannon Divergence
Prediction Confidence	Measures the uncertainty in model predictions.	Provides insights into model confidence and potential for errors.	Prediction entropy, Model Confidence Variance
Data Quality Metrics	Metrics that track the integrity and consistency of the input data.	Ensures that the data used for training and inference is valid.	Missing Values Rate, Data Completeness Score
Latency and Throughput	Metrics related to the time taken for predictions and the volume of requests processed.	Monitors the operational efficiency of the system.	Response Time, Prediction Volume

Challenges in Implementing Observability for Non-Deterministic Workloads

The implementation of observability presents substantial technical, operational, and ethical obstacles for controlling systems powered by AI or machine learning, even though these components serve as essential management tools for nondeterministic systems. The hurdles preventing observability arise because systems demonstrate a complex nature as well as due to data restrictions, infrastructure limitations, and compliance requirements.

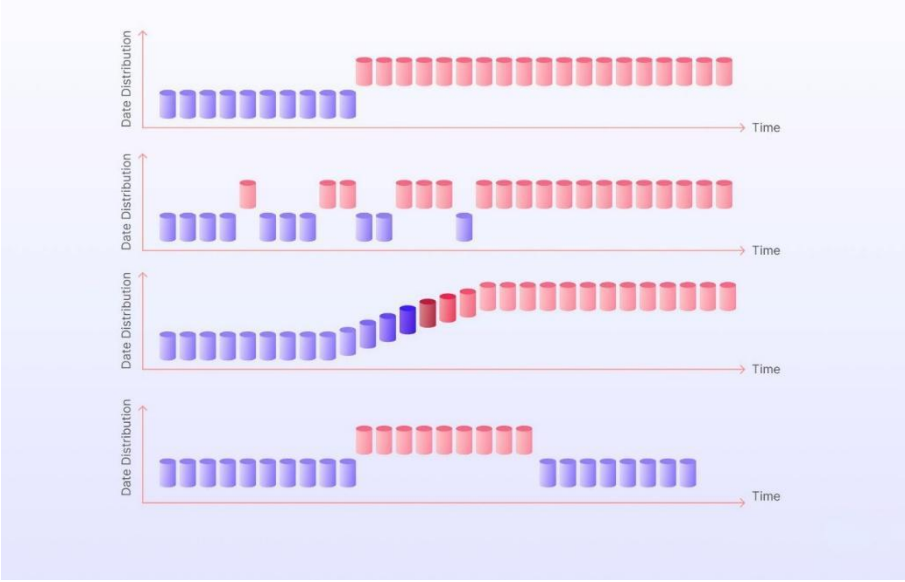

Image 2: Model Drift Over Time (Graphical Representation) *[10]*

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

Real-Time Model Drift Detection

The major obstacle in practice involves identifying model drift while it happens in real time. Model drift differs from typical system failures because the symptoms include predictive performance degradation in place of crashes or performance issues. The data environment changes over time, so the model based on previous data points fails to correctly predict new inputs. The process of identifying drift requires active statistical checks between current inputs and training data distributions with access to ground truth labels needed for outcome evaluations. Real-world applications present challenges with these labels since they encounter delays and incompleteness as well as lack of availability, particularly within healthcare and finance domains and recommendation systems. When teams lack access to ground truth feedback, they must use prediction entropy and confidence intervals as proxies, but these indicators sometimes produce inaccurate results regarding performance issues [11].

Data Privacy and Regulatory Compliance

The practice of observability requires data capture along with the logging of model inputs and outputs and intermediate feature data while handling both personal information and other sensitive data categories. GDPR in Europe and the CCPA in the United States, together with other similar state regulations, demand rigorous limitations on data handling, including collection and usage activities. The requirement for visibility creates opposition with regulatory compliance guidelines. Observability frameworks need to adopt privacy-preserving methods such as anonymized data, aggregated data, redaction, and differential privacy protection to reduce data privacy risks. The implementation of privacy-preserving techniques leads to increased framework complexity and decreases the level of both the observable details and the accuracy that emerges from observability pipelines.

Resource Constraints and Latency Overhead

When implementing observability at scale, it leads to significant non-subtle resource utilization among computing resources and storage systems. The operations of data pipeline monitoring combined with drift metric calculation and SHAP or LIME explanation generation, together with prediction history storage, require measurable amounts of CPU time and memory increases along with added latency. Additional computations for production workflows at critical speeds, such as fraud discovery and autonomous systems and recommendation platforms, reduce system latency, which harms user engagement and product stability. Organizations can address this situation through different approaches that consist of sampling methods combined with asynchronous monitoring technologies and multiple alert levels for efficient resource allocation [12].

Tooling Fragmentation and Integration Complexity

The development of the AI observability toolset remains ongoing as different monitoring components, such as Prometheus metrics and Arize AI model tracking, WhyLabs data profiling, and AI drift detection, operate independently from each other. The diverse set of requirements between different monitoring tools creates difficulties for integrating their APIs, data formats, logging schemas, and visualization interfaces into a single observability framework. The absence of standardized practices causes project teams to devote extensive work to piecing components together and maintaining data coherence between tools while creating effective monitoring dashboards and alarms.

The indisputable value of observability for managing non-deterministic AI/ML systems demands careful evaluation of trade-offs together with compliance requirements and technical limitations, as well as ecosystem fragmentation, to achieve successful implementation. Trustworthy and transparent AI systems, together with their resilience, depend heavily on addressing these mentioned challenges [13].

Table 2: Challenges in Implementing Observability for Non-Deterministic AI/ML Workloads
It outlines key challenges in establishing observability for non-deterministic AI/ML workloads, including issues with drift detection, data privacy, performance trade-offs, and tooling fragmentation.

Challenge	Description	Impact on Observability
Model Drift Detection	Detecting gradual performance degradation in models over time.	Difficult to detect without labeled data, making it harder to proactively address.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

Data Privacy and Security	Observability requires logging data that may include sensitive user information.	Compliance with regulations (GDPR) requires extra precautions, impacting visibility.
Resource Constraints and Latency	Continuous monitoring and drift detection may slow down inference pipelines.	Potential impact on real-time inference, requiring trade-offs between visibility and speed.
Tooling Fragmentation	Multiple observability tools often do not integrate seamlessly.	Integration and alignment of different tools can require significant effort and resources.

Tooling Ecosystem for AI/ML Observability

Business needs for robust AI/ML observability tools emerged to tackle the rising complexity of ML systems operating in unpredictable production environments. The behavior of AI/ML systems goes beyond conventional software since they generate unpredictable results from data dependencies alongside model drifts, probabilistic predictions, and the need for constant model retraining. Observability within ML systems needs to include three main aspects: system performance measurements alongside data quality evaluation and model conduct assessment and a history tracking capability [14].

Infrastructure-Level Observability Tools

Below the observability stack is where traditionally infrastructure monitoring tools such as Prometheus and Grafana sit in the vast adoption, collecting and visualizing time-series metrics at the bottom of the observability stack. These tools monitor operational indicators like CPU usage, memory consumption, disk I/O, and network latency. They are (almost) not ML-specific but are crucial to ensure the survival and availability of ML systems and services. Custom exporters can be used to extend Prometheus and to track metrics like inference latency, throughput, prediction volume, and failure rates specific to ML. And subsequently, Grafana allows you to create custom dashboards that show your metrics over time and gives your teams a high-level view of their deployed systems.

ML-Specific Observability Platforms

A new generation of observability platforms has come up to overcome machine learning nuances. Arize AI is a top-performing ML observability tool built from commercial needs, so it is designed to solve your requirements. This gives end-to-end visibility into deployed models with monitoring of Accuracy, precision, recall, and AUC. Built in with its ability to detect model drift, data drift, and bias, Arize also excels in real-time alerting and root cause analysis. It is highly scalable, supports multiple models across environments, and is integrated with common MLOps pipelines [15].

Fiddler AI is another relevant platform aimed at model explainability, fairness, and traceability. By exposing feature-level attributions with explainability techniques such as SHAP values, Fiddler allows practices to provide us with an insight into why a model would make such a prediction. This is especially applicable in regulation industries where arguing and understanding model decisions is critical. Finally, Fiddler comes with batch and real-time inference environments, thus providing clarification to all prediction lifecycles.

There are also open-source alternatives that have gained popularity because they are flexible and cheap. WhyLabs, built around the Whylogs library, offers ways to quietly but confidentially log datasets and model outputs. Whylogs captures statistical summaries and metadata that can be used to detect anomalies, monitor feature distributions, and flag data drift. They have designed the solution for large-scale production and privacy first, which makes it suitable for applications like finance and healthcare due to its ability to work with sensitive information [16].

Just like Evidently, AI is an open-source, user-friendly tool to generate visual dashboards and interactive reports on features, labels, data quality metrics, and model performance over time. Its no-code interface provides data scientists and ML engineers the ability to plug their model inputs and outputs to get detailed analysis with no deep integration to the external platform. Specifically, it is especially helpful during the validation, deployment, and post-deployment phases of the ML lifecycle.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

Experiment Tracking and Reproducibility Tools

For observability during the whole ML life cycle, we also need tools like experiment tracking, model versioning, and reproducibility. It is a widely used open-source platform that allows teams to monitor experiments, store model artifacts, and manage lifecycle stages like training, staging, and production. It also works with the usual ML libraries and deployment tools seamlessly and provides observability from development to deployment [17].

As with Git, DVC (Data Version Control) manages datasets and training pipelines as versioned artifacts using the same facilities that Git provides to manage code. Proper reproducibility of experiments ensures collaboration among team members and facilitates the standardization of experiments to be carried out in the future.

Data Lineage and Pipeline Transparency

At last, the application of data lineage tools like Amundsen and OpenLineage strengthens the transparency in AI systems. These platforms offer platforms for tracing data in the flow of complex pipelines, which helps in deciding the places of data origin, what is transformed, and to which models it goes. Audit, debug, and comply with regulation, especially for industries that need strict data governance and require this visibility [18].

Overall, the area of observability for AI / ML is quite dynamic, and there are many different tools in the ecosystem. With the help of the correct set of infrastructure-level tools, model-specific platforms, open-source libraries, and lineage tracking systems, these organizations can build end-to-end observability stacks as per their ML operations. In addition to increasing system reliability, these tools help to increase trust, accountability, and performance of AI-driven decision-making.

Table 3: Common Observability Tools for AI/ML Systems
It presents widely used tools for AI/ML observability, covering monitoring, visualization, explainability, and model tracking to support reliable and interpretable system performance.

Tool	Type	Key Features	Use Case
Prometheus	Infrastructure Monitoring	Collects time-series data on system performance.	Tracks infrastructure metrics like CPU, memory, and uptime.
Grafana	Data Visualization	Open-source tool for visualizing time-series data from Prometheus.	Creates dashboards for real-time visualization of model performance.
Arize AI	AI/ML Observability Platform	Real-time monitoring of model performance, drift, and bias.	Detects model drift, monitors prediction quality, and identifies biases.
Fiddler AI	Model Explainability & Fairness	Provides explainability, fairness checks, and model debugging.	Helps with debugging and ensuring model fairness during production.
MLflow	Model Tracking & Versioning	Tracks experiments, model versions, and provides reproducibility.	Ensures traceability of ML model versions and parameters.

Real-World Case Studies

As a result, a number of technology companies who are leading the pack have developed bespoke solutions and platforms that integrate tightly in observability across the ML lifecycle to address the challenges that arise with observability in nondeterministic AI/ML systems. Finally, these real-world implementations provide insights into the ability to approach robust monitoring, data lineage, and explainability at scale.

Uber: Michelangelo Platform

Michelangelo is Uber's complete ML platform, which was built to be one of the most comprehensive ML platforms in the industry. The observability of every stage of the machine learning pipeline from ingestion of data and feature engineering to deployment and retraining is embedded into this platform. Uber uses Kafka streams to continuously read the feature data and result from the prediction in order to continuously serve the real-time model behavior insight. It monitors all active models and provides metrics like how much latency, accuracy, and prediction volume there are

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

in them. Uber's airflow dag orchestrates the automated pipeline of anomaly detection pipelines and retrain workflows so that Uber can be proactive of model drift or quality data issues. Model metadata, lineage, and usage statistics are centralized by Michelangelo to provide better traceability, reproducibility, and accountability on thousands of models deployed everywhere on the globe [19].

Airbnb: Zipline and Feature Observability

Airbnb's main avenue for observability is through its Zipline system, which concerns itself with feature engineering. Unlike with many other approaches,s where features can be used in a variety of ad hoc ways, Zipline takes the first-class view of features. It provides central storage, version control, and drift detection of feature sets. Airbnb engineers can also monitor what has changed over time in feature distributions, link them to fluctuations in model performance, and establish alerts triggered by a large deviation. The ability to see this level of visibility provides for a fine root cause analysis, and if the degradation of performance occurs, early intervention to reduce the time-to-detection and resolution by an order of magnitude.

Netflix: Observability and Experimentation

Monitoring, explainability, and experimentation are combined by Netflix when it comes to observability. It combines model outputs interpretation through tools such as SHAP with A/B testing on the real world to understand what model changes cause. To illustrate, teams can see not only the performance metrics, such as click-through rates or engagement scores, but also the explanatory factors that are driving those predictions when a recommendation model is updated. Through their dual-layered approach, Netflix can validate model updates through business outcomes and ensure that improvements align with strategic goals.

These case studies together demonstrate how observability can be operationalized in this type of complex machine learning use case by leveraging automated data pipelines, interpretability tools, and real-time metrics together.

Best Practices and Recommendations

The strategic planning and tactical execution of implementing observability in nondeterministic AI/ML systems is very important. Since AI/ML systems are a matter of nature, they are very complex to tackle due to dependencies and cause the unpredictability of behavior, as well as the variability in the behavior of the system itself. Therefore, it becomes necessary for organizations to implement good practices that would facilitate a proper monitoring and analysis of these systems. Below are some key recommendations for achieving robust observability in AI/ML environments:

Granular Metadata Logging

Granular metadata logging is the basis of effective observability. It is crucial to get detailed logs while they pass through each stage of the inference request so they can be analyzed post hoc. That includes the form of basic model information, e.g., model version, timestamp, and input features, but also the form of prediction metadata: prediction confidence, output probabilities, and any associated model parameters. This level of logging allows teams to see when an issue first appeared, helps them understand the root cause of performance degradation, spot potential bias, and debug to the code and data that influences model behavior [20].

Automated Drift Detection

As their name implies, data drift and model drift are the major problems in nondeterministic AI/ML systems. To track perturbations such as changes in the distribution of features and labels with time, automated drift detection on features and labels is an essential technique based on statistical techniques. One can use key metrics such as the Population Stability Index (PSI) or Kullback-Leibler Divergence for detecting changes in the data characteristics. Teams can proactively respond to significant deviations by setting up automated alerts for such deviations so that performance degradation goes unnoticed in production or model behavior becomes unexpected.

Service Level Objectives (SLOs) for ML Models

SLOs are an essential tool for keeping an eye on the work that is getting fulfilled by ML models to make sure that these models are delivering what's required. Previous SLOs should include clearly defined targets for latency, accuracy, drift tolerance, and other such critical performance metrics. Teams can monitor these, as well as traditional metrics of infrastructure (e.g., CPU, RAM usage) in a balanced way (Model health and System health). By reviewing and revising these SLOs based on the changing business goal and characteristics of the data, these SLOs retain their relevance to the objective for which the model was built.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

Explainability Integration

Also essential for being able to use model explainability tools residing within observability frameworks is that they are embeddable. Some dashboard and alerting system techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can be woven into the dashboard to help you to understand what specific features did predict. Explainability is critical when people have to understand and verify model decisions and their justification in high-stakes applications such as healthcare, finance, or autonomous systems [21].

Privacy-Aware Observability

This is important given that the importance of data privacy regulations such as the General Data Protection Regulation (GDPR) continues to rise. Data anonymization or differential privacy or other privacy-preserving logging techniques are a must, and data minimization and role-based access control are important. As an observation, these practices make sure that observability doesn't disrupt user privacy while also maintaining compliance if such data is involved, for example.

End-to-End Pipeline Tracing

End-to-end pipeline tracing is required to get a real-life picture of performance for AI/ML systems. OpenTelemetry is the solution/tool that helps teams trace the data and predictions across multiple services in a distributed ML service. With this end-to-end visibility, we can see where data flows in and out of ingestion, through transformation all the way into the predictions given by the model. It also makes it easier to monitor how such model updates affect the system as time progresses.

Table 4: Best Practices for Implementing AI/ML Observability
It highlights best practices for effective AI/ML observability, emphasizing metadata logging, drift detection, performance objectives, and explainability to enhance reliability and transparency.

Best Practice	Description	Benefit
Granular Metadata Logging	Capture detailed information such as model version, input features, and prediction metadata.	Enables deeper post-hoc analysis and helps track model behavior over time.
Automated Drift Detection	Use statistical methods to monitor feature and label distributions and trigger alerts for significant changes.	Helps detect issues early and improves model stability.
Service Level Objectives (SLOs)	Set targets for model latency, accuracy, and drift tolerance, and monitor them alongside infrastructure metrics.	Ensures models meet performance standards and maintain reliability.
Explainability Integration	Integrate tools like SHAP and LIME to provide insights into model decisions.	Improves accountability, debugging, and trust in model decisions.

Future Directions in Observability

Observability itself will evolve as more and more autonomous and adaptive AI systems come online. Causal observability is one emerging area that expands from correlation-based monitoring to discover causal relationships between the system inputs and behaviors. This would make it possible for more accurate root cause analysis and intervention strategies.

Another area where machine learning is promising is the use of artificial intelligence to improve observability systems. For instance, anomaly detection algorithms can find unusual patterns in logs or metrics, clustering techniques can group similar incidents, and reinforcement learning can optimize alerting policies [22].

Similar attention is being paid to federated observability, especially when deployed models are spread across decentralized devices. For example, in this case, safety comes with secure aggregation protocols as well as privacy-preserving metrics.

CONCLUSION

There is no longer luxury in observing AI/ML systems since these systems often exhibit non-deterministic behaviors, and it's mandatory. With more and more importance placed upon their use in decisions, safety, and business processes,

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

the importance and reliability of these systems must be ensured. When the context of observability is high velocity, managing observability requires an in-depth, holistic approach encompassing data, model, infrastructure health, and user interaction. At the same time, new tools, new culture, and ever-increasing investment in system design are required.

While challenges persist (e.g., in tooling fragmentation and data privacy concerns), the progress in this space is an area of hope. Proactive organizations that make investments toward observability are more likely to get insights early to detect the issues and improve their model performance, regulate them, or simply build trustworthy AI systems. Observability is the system's mirror and compass it will act as a mirror of system behavior today and a compass for responsible development tomorrow.

REFERENCES

1. Usman, M. (2022). A survey on observability of distributed edge. *IEEE Xplore*. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9837035
2. Sculley, D., et al. (2015). *Hidden Technical Debt in Machine Learning Systems*. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NeurIPS), 2503–2511. https://papers.nips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html
3. Ruan, J., Liang, G., Zhao, J., Zhao, H., Qiu, J., Wen, F., & Dong, Z. Y. (2023). Deep learning for cybersecurity in smart grids: Review and perspectives. *Energy Conversion and Economics*. https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/enc2.12091
4. Schröder, T., & Schulz, M. (2022). Monitoring machine learning models: A categorization of challenges and methods. *Data Science and Management, 5*(3). https://www.researchgate.net/publication/362422976_Monitoring_machine_learning_models_A_categorization_of_challenges_and_methods
5. Thomas, P. (2024,). *Enhancing observability in machine learning with OpenTelemetry: InsightfulAI update*. https://dev.to/craftedwithintent/enhancing-observability-in-machine-learning-with-opentelemetry-insightfulai-update-55p6
6. Retinraj, P. D. (2022, December 31). *Designing a model monitoring and observability system*. *AI Science*. https://medium.com/ai-science/designing-a-model-monitoring-and-observability-system-d28cd8735c2a
7. Gama, J., Medas, P., Castillo, G., & Rodrigues, P. P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, São Luís, Maranhão, Brazil, September 29 – October 1, 2004, Proceedings* (pp. 286–295). Springer. https://www.researchgate.net/publication/220974771_Learning_with_Drift_Detection
8. Koponen, E. (2021, December 2). Observability in production: Monitoring data drift with WhyLabs and Valohai. *Valohai*. https://valohai.com/blog/monitoring-data-drift/
9. Polachowska, K. (2022, September 20). Why do we need explainable artificial intelligence? *ReasonField Lab*. https://softwaremill.com/why-do-we-need-explainable-artificial-intelligence/
10. Encord. (2024, January 4). *Model drift: Best practices to improve ML model performance*. Encord Blog. https://encord.com/blog/model-drift-best-practices/
11. Suárez-Cetrulo, A. L., Quintana, D., & Cervantes, A. (2022). A survey on machine learning for recurring concept drifting data streams. *Expert Systems with Applications, 213*, 118934. https://www.researchgate.net/publication/364148850_A_survey_on_machine_learning_for_recurring_concept_drifting_data_streams
12. Amershi, S., Inkpen, K., Teevan, J., & Kikin-Gil, R. (2019, April). *Guidelines for human-AI interaction*. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery. https://www.researchgate.net/publication/332742200_Guidelines_for_Human-AI_Interaction
13. Gilpin, L. H., et al. (2018). *Explaining Explanations: An Overview of Interpretability of Machine Learning*. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. https://arxiv.org/abs/1806.00069

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://www.ijetrm.com/

14. Sharifani, K., & Amini, M. (2023). Machine learning and deep learning: A review of methods and applications. *MahamGostar Research Group*. https://www.researchgate.net/publication/371011515_Machine_Learning_and_Deep_Learning_A_Review_of_Methods_and_Applications
15. Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2017, May). Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 1723–1726). Association for Computing Machinery. https://dl.acm.org/doi/10.1145/3035918.3054782
16. Joulin, A., et al. (2017). *Bag of Tricks for Efficient Text Classification*. arXiv preprint arXiv:1607.01759. https://arxiv.org/abs/1607.01759
17. Koh, P. W., & Liang, P. (2017). *Understanding Black-box Predictions via Influence Functions*. In Proceedings of the 34th International Conference on Machine Learning, 70, 1885–1894. https://proceedings.mlr.press/v70/koh17a.html
18. Lundberg, S. M., & Lee, S. I. (2017). *A Unified Approach to Interpreting Model Predictions*. Advances in Neural Information Processing Systems (NeurIPS), 30. https://papers.nips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
19. Pugliese, R., Regondi, S., & Marini, R. (2021). Machine learning-based approach: Global trends, research directions, and regulatory standpoints. *Data Science and Management, 4*, 19–29. https://www.sciencedirect.com/science/article/pii/S2666764921000485
20. Ribeiro, M. T. (2016). "Why should I trust you?": Explaining the predictions of any classifier. *ACM Digital Library*. https://dl.acm.org/doi/10.1145/2939672.2939778
21. Tian, H., Zhu, T., Liu, W., & Zhou, W. (2022). Image fairness in deep learning: Problems, models, and challenges. *Neural Computing and Applications, 34*(15), 12875–12893. https://dl.acm.org/doi/10.1007/s00521-022-07136-1
22. Tjoa, E., & Guan, C. (2021). A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems, 32*(11), 4793–4813. https://pubmed.ncbi.nlm.nih.gov/33079674/