International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I11P107

Volume 72 Issue 11, 63-71, November 2024


*Original Article*

# Challenges, Solutions, and Best Practices in Post-Deployment Monitoring of Machine Learning Models

Surabhi Bhargava[1], Shubham Singhal[2]

[1]*Senior Machine Learning Engineer, Adobe.*
[2]*Software Engineer, Adobe.*

[1]*Corresponding Author : surabhi9b@gmail.com*

Received: 22 September 2024 Revised: 25 October 2024 Accepted: 14 November 2024 Published: 30 November 2024

***Abstract*** *- In production environments, machine learning models often encounter data and operational conditions that differ significantly from the training environment. These differences lead to various challenges, such as data drift, concept drift, harmful feedback loops, adversarial attacks, model failures, and potential biases that may emerge in real-world applications. Model interpretability also becomes crucial in these environments, as understanding how models make decisions is necessary for debugging, trust-building, and mitigating any inadvertent biases that could lead to unfair outcomes. This paper explores these challenges in depth, presenting effective strategies for handling them. Drawing from industry practices and research insights, the paper outlines key solutions such as dynamic retraining, versioning, adversarial training, robust monitoring, and fairness-aware model evaluation to ensure continued model performance and equity post-deployment.*

***Keywords*** *- MLOps, Data and Concept drift, Model integrity, Adversarial attacks, Feedback loop.*

## 1. Introduction

Once Machine Learning (ML) models are deployed in production, they are exposed to complex, dynamic environments that often differ significantly from the controlled conditions of their development stage. While extensive research has been conducted to optimize model performance in training environments, relatively less attention has been given to the unique set of challenges that arise post-deployment. In production, models frequently encounter issues such as data drift, concept drift, harmful feedback loops, adversarial attacks, and failures in the ML pipeline. These challenges can result in performance degradation and security vulnerabilities, making implementing robust monitoring, retraining mechanisms, and proactive defense strategies essential.

This paper addresses the gap between training-focused research and the needs of production environments by examining post-deployment challenges. Existing studies have typically focused on enhancing model accuracy and robustness within controlled settings. This paper focuses on maintaining model reliability in real-world operational environments. It provides a comprehensive analysis of common pitfalls in post-deployment monitoring, performance degradation, and security vulnerabilities, as well as effective solutions derived from industry practices and recent advancements in research. This paper contributes novel insights and actionable recommendations for ensuring deployed ML models' long-term robustness and security by bridging the gap between academic research and industry needs.

## 2. Common Points of Failures

### *2.1. Data and Concept Drift*

*2.1.1. Data Drift*

Data drift [1, 2] occurs when the input data distribution used to train a machine learning model diverges from the distribution of the input data that the model encounters during deployment. This difference can cause a reduction in the model's performance, leading to lower accuracy and less reliable predictions or decisions.

Mathematically, data drift can be expressed as:

$$P(x|y) \neq P(x|y')$$

where P(x|y) denotes the probability distribution of the input data (x) conditioned on the output data (y), and P(x|y') represents the probability distribution of the input data given the new output data (y') for the deployed model.

*Causes of Data Drift*

- *Changes in Data Sources*: Variability in data collection methods, such as changes in sensors, equipment, or external data providers, can introduce data drift. Example: A retail company might change its data provider for customer demographics, which could lead to inconsistencies in customer segmentation data.

- *User Behavior Changes*: User preferences can evolve over time, affecting applications like recommendation systems, which rely on historical user behavior to make predictions. Example: A streaming service's recommendation model may start underperforming if user preferences shift towards a new genre or type of content not adequately represented in the training data.

- *Seasonality and Trends*: Cyclical trends in data, such as seasonal patterns, can lead to periodic data drift.

- Example: In an e-commerce setting, purchase behaviors during holiday seasons may differ significantly from the rest of the year, causing fluctuations in data that models may not be equipped to handle.

- *Market Dynamics*: For financial models, sudden market shifts, economic events, or new regulations can cause the data distribution to drift. Example: During an economic recession, credit risk models may underperform due to changes in spending behavior and default rates.

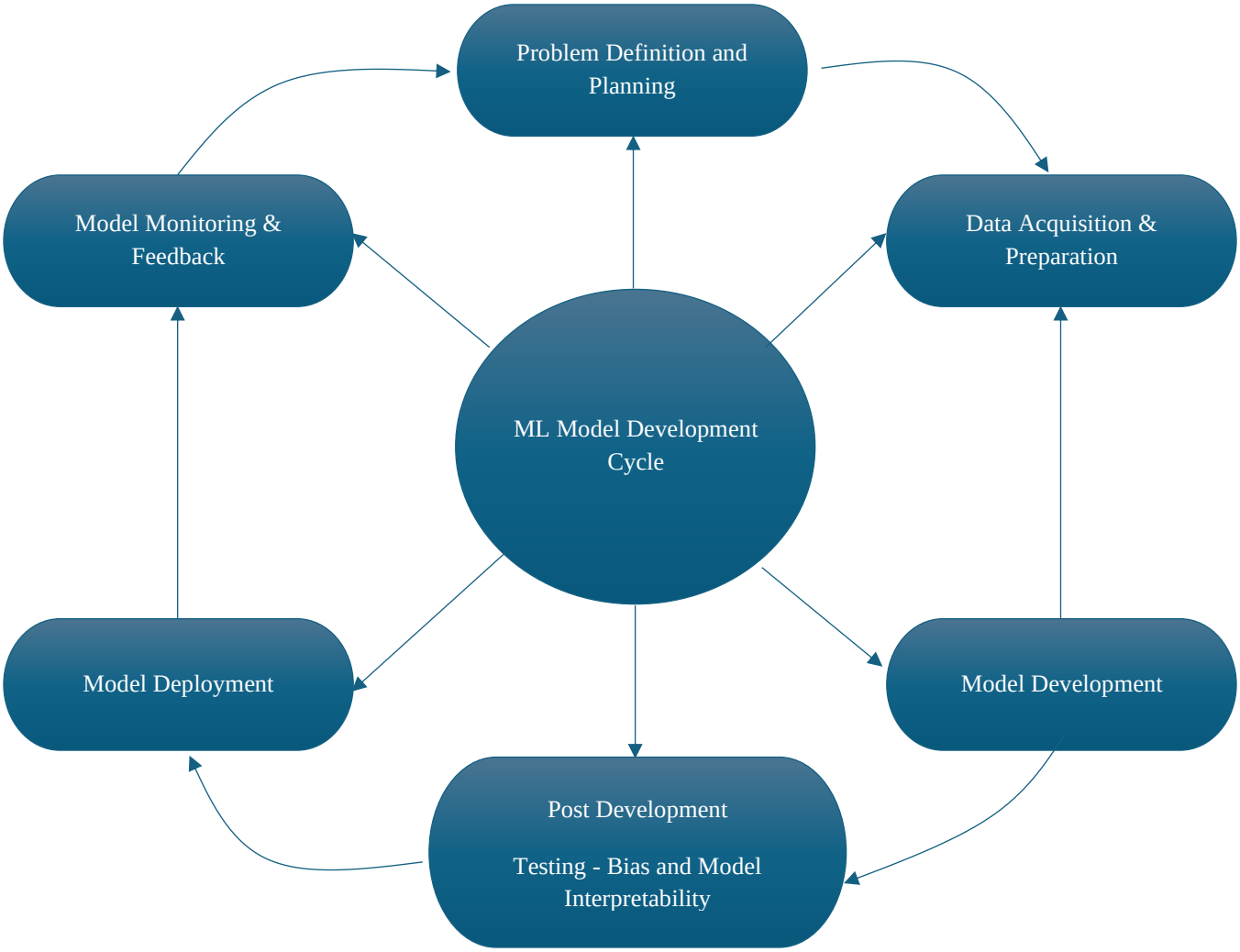


**Fig. 1 ML model development lifecycle**

*2.1.2. Concept Drift*

Concept drift [1, 2] occurs when the underlying relationship between a model's input and output data evolves over time. In Concept Drift, the model is trying to predict changes over time. However, the model continues to operate as if no changes have occurred. Consequently, the patterns it learned during training are no longer accurate.

Mathematically, it can be represented as follows:

$$P_{t1}(Y|X) \neq P_{t2}(Y|X)$$

*Examples:*

- *Spam email detection*: Assume a model has been trained to detect spam emails using a large dataset. Over time, spam emails evolve, becoming more sophisticated and resembling legitimate emails. This is an example of concept drift, where the characteristics that define a spam email have changed, causing the model's understanding of "spam" to become outdated.
- *Grocery sales prediction*: Assume a model has been trained to predict the grocery sale pattern over many

years. The pandemic introduced drastic changes in grocery sales patterns that disrupted existing predictive models. People began stockpiling essentials, shifting preferences towards cooking ingredients, and increasingly relying on online shopping, while supply chain disruptions impacted availability. These changes altered demand patterns, making pre-pandemic data and trends less relevant.

### *2.2. Pipeline Integrity Issues*

Feature pipelines typically involve various steps such as data preprocessing, feature extraction, one or more ML models and external APIs. A change in any upstream or downstream component, such as feature scaling, input data formats, or number of input dimensions, can cause errors in model predictions. For instance, a model expecting RGB images but receiving grayscale images may fail without obvious symptoms until errors accumulate.

### *2.3. Model Interpretability Failures*

Model interpretability is crucial for understanding how and why a model makes certain predictions, especially in high-stakes fields like healthcare, finance, or criminal justice [3]. When a model is not interpretable, it can obscure critical issues or errors in its predictions.

Lack of interpretability can also limit trust from stakeholders and users, who may be unwilling to adopt or rely on a model they do not understand. Complex models like deep neural networks are “black boxes” because their internal decision-making processes are challenging to interpret. When these models make incorrect predictions, it is difficult to pinpoint the cause, complicating debugging, retraining, and correction efforts.

### *2.4. Bias and Fairness Failures*

Bias in machine learning can emerge when the training data is not representative of the full population or contains inherent biases. When unchecked, these biases can lead to unfair and potentially harmful outcomes, especially in applications that impact people’s lives. Bias and fairness-related failures include:

- *Historical Bias in Training Data*: If a model is trained on biased historical data, it will likely perpetuate or even amplify those biases in its predictions. For example, if a hiring algorithm is trained on historical recruitment data that reflects gender or racial biases, the model may favour certain groups unfairly.
- *Sampling Bias*: When data collection processes unintentionally exclude certain population segments, the model may become less accurate for those groups. This can result in lower performance for underrepresented groups and exacerbate inequalities, particularly in healthcare or credit scoring sectors.
- *Unfair Treatment of Sensitive Attributes*: Machine learning models sometimes make decisions based on sensitive attributes like age, gender, or race, even if these features are not directly included in the model. Proxy variables correlating with these attributes may still influence model predictions, resulting in discriminatory outcomes. For instance, a model predicting recidivism rates may inadvertently assign higher risk scores to certain demographic groups due to biases embedded in proxy features. [4]

### 2.*5. User Feedback and Feedback Loops*

User feedback is essential to refining and improving machine learning models in production. However, it can also introduce complexities and unintended consequences, especially when feedback loops occur [5, 6, 7]. A feedback loop arises when the model continuously learns from user interactions, which can introduce biases or cause it to deviate from its original generalization objective.

Some common challenges that arise are:

#### *2.5.1. Bias Amplification and Model Overfitting*

User feedback mechanisms can unintentionally lead to bias amplification, where models become overfitted to the preferences of a small subset of active users. As these users provide frequent feedback, the model adjusts its parameters disproportionately toward their specific preferences, causing a shift in decision boundaries.

Over-optimizing to a localized section of the feature space compromises the model’s generalization across the overall distribution, reducing its robustness in serving a broader population. The feedback mechanism inherently narrows the model’s capacity to adapt to unseen data, increasing the risk of biased predictions and reduced fairness.

#### *2.5.2. Feedback Fatigue and Limited Participation*

Another challenge is feedback fatigue, where users tire of providing constant feedback, especially if they do not see immediate improvements. In many cases, only a small percentage of users actively provide feedback, which can skew the model’s learning toward those few users.

#### *2.5.3. Difficulty in Breaking Feedback Loops*

Once a feedback loop starts, it can be difficult to break or reverse. For example, if a model has adjusted heavily based on feedback from a specific group of users, reverting those changes without negatively impacting model performance can be challenging.

### *2.6. Adversarial Attacks*

Adversarial attacks [8, 9] are a significant risk for deployed Machine Learning (ML) models, particularly as they are increasingly used in critical applications such as healthcare, autonomous driving, finance, and security systems. These attacks exploit vulnerabilities in the models by intentionally crafting inputs that cause the model to make incorrect or undesirable predictions. Adversarial attacks pose

a unique challenge because they can be subtle, difficult to detect, and have serious consequences if successful.

Some common types of Adversarial attacks are listed below.

*2.6.1. Evasion Attacks*

Evasion attacks [10] exploit the sensitivity of machine learning models by introducing carefully crafted perturbations to input data, resulting in misclassifications while remaining imperceptible to humans. These adversarial perturbations can be small but strategically designed to cause significant deviations in the model's predictions.

For instance, slight modifications to a stop sign image in image classification may lead a self-driving car's vision system to misclassify it, potentially resulting in hazardous outcomes like traffic accidents. The subtle nature of these perturbations allows them to bypass human oversight, making evasion attacks particularly dangerous and challenging to detect in real-time systems.

*2.6.2. Poisoning Attacks*

Poisoning attacks [11] occur when an adversary injects malicious data into the training set, deliberately contaminating the learning process of a machine learning model. This manipulation influences the model to behave in ways that benefit the attacker.

For example, in a spam detection system, an attacker could introduce adversarial spam emails labeled as non-spam, thereby corrupting the decision boundary and reducing the model's effectiveness in filtering future spam.

These attacks are particularly detrimental to systems that rely on continuous learning or frequent retraining, as they regularly incorporate new data into the model. In the case of fraud detection, an attacker might inject fraudulent transaction data labeled as legitimate, compromising the model's ability to identify fraudulent behavior accurately and leading to systemic vulnerabilities.

*2.6.3. Model Inversion Attacks*

Model inversion attacks [12] refer to adversarial activities to reverse-engineer sensitive training data by systematically querying a machine learning model and analyzing its outputs. Through this process, attackers can infer private details from the training set, exposing information such as personal data or patterns related to specific individuals.

These attacks raise significant privacy concerns, particularly when models are trained on sensitive datasets, such as medical records, financial data, or Personally Identifiable Information (PII). For example, in facial recognition systems, an adversary could leverage the model's outputs to reconstruct visual representations of individuals from the training data, potentially leading to severe privacy breaches and unauthorized data exposure.

# 3. Solutions and Best Practices

## *3.1. Handling Data and Concept Drift*

*3.1.1. Regular Monitoring of Data Distributions*

Continuous monitoring of training data distributions and production data is essential for detecting data and concept drift. Statistical techniques, such as the Population Stability Index (PSI) or entropy-based drift detection, can identify significant deviations in feature distributions or prediction patterns from the original training data. Early detection of such drifts ensures timely corrective actions and maintains model reliability in production environments.

*3.1.2. Dynamic Retraining*

As data and feature relationships evolve, dynamic retraining becomes critical to model adaptation. Establishing automated pipelines that periodically incorporate fresh production data into the retraining process helps models evolve with real-world changes. Retraining cycles can be triggered based on predefined drift metrics, enabling the model to stay aligned with the current data distribution and maintain optimal performance. [1, 2]

## *3.2. Ensuring Pipeline Integrity to Prevent ML Pipeline Failures*

*3.2.1. Implementing Contracts for Model Inputs and Outputs*

Defining a strict contract for the expected format and input data type helps ensure that any changes in upstream components are flagged before they cause model failures.

*3.2.2. Data and Model Versioning*

Versioning the entire pipeline, from data preprocessing, feature engineering and model training, enables teams to track changes, pinpoint performance degradation, and revert to previous configurations when necessary.

*3.2.3. Automated Unit and Integration Testing*

Unit testing of individual components and integration testing are used to verify the seamless interaction between these components and validate the pipeline's overall functionality and reliability. [13]

## *3.3. Improving Model Interpretability*

Improving model interpretability ensures that Machine Learning (ML) models are transparent, trustworthy, and accountable, particularly in high-risk domains where model predictions can significantly impact individuals and society. Below are strategies for improving the interpretability of ML models, especially in complex systems like deep learning -

*3.3.1. Use of Interpretable Models*

Choosing inherently interpretable models can simplify the process in situations where interpretability is a top priority. Models such as decision trees, linear regression, and logistic regression are transparent and allow an easy understanding of how input features contribute to the final predictions.

*3.3.2. Model-Agnostic Interpretability Methods*

For more complex, black-box models like deep neural networks, model-agnostic interpretability techniques provide a way to explain predictions without altering the underlying model architecture. Two popular approaches are:

*3.3.3. LIME (Local Interpretable Model-agnostic Explanations)*

LIME approximates complex models with interpretable surrogate models by perturbing the input data and observing the impact on model predictions. This provides local explanations for individual predictions, making understanding why a model behaves a certain way for a specific input easier. [14]

*3.3.4. SHAP (SHapley Additive exPlanations)*

SHAP assigns importance values to each feature in a model by computing Shapley values based on cooperative game theory. SHAP can explain how each feature influences a prediction, allowing for consistent and comparative feature importance rankings across different inputs. [15]

*3.3.5. Visualization Techniques*

Visualization is a powerful tool for interpreting and understanding how models make decisions, especially for deep learning models. Some of the common techniques for visualization are mentioned below:

- *Activation Maps*: In Convolutional Neural Networks (CNNs), activation maps can highlight which parts of the input image are most influential in the model's decision-making process. This helps reveal which features are considered most important in image classification tasks. [16]
- *Saliency Maps:* These maps indicate which parts of the input data (e.g., pixels in an image or words in text) are most sensitive to changes in the output, showing the influence of specific features on model predictions.[17]
- *Partial Dependence Plots (PDPs)*: PDPs show how the model's output changes as a single feature varies, providing insight into the relationship between input features and model predictions.[18]

***3.4. Mitigating Bias and Ensure Fairness***

Mitigating bias and ensuring fairness in machine learning models is essential to avoid discriminatory outcomes, particularly in sensitive applications. Here are key techniques [19, 20, 21, 22, 23] to mitigate biases and ensure fairness.

*3.4.1. Preprocessing Techniques*

- *Reweighting and Resampling:* Adjust data to ensure underrepresented groups are sufficiently represented (e.g., oversampling minority groups).
- *Data Augmentation:* Generate synthetic data to balance representation.
- *Fair Representation Learning:* Transform data to reduce correlations with sensitive attributes.

*3.4.2. In-Training Techniques*

- *Fairness Constraints in Training:* Introduce fairness constraints during model training to ensure equitable outcomes (e.g., equal opportunity, demographic parity).
- *Adversarial Debiasing*: Use adversarial networks to prevent models from learning biased patterns.
- *Regularization for Fairness:* Add fairness terms to the loss function to penalize biased outcomes during training.

*3.4.3. Post-Processing Techniques*

- *Equalized Odds and Calibration:* Adjust predictions post-training to ensure fairness across groups (e.g., equal true positive rates).
- *Re-ranking:* Modify the ranking of predictions to ensure equitable representation of underrepresented groups.
- *Bias Correction:* Adjust outputs to remove correlations between sensitive attributes and predictions.

*3.4.4. Fairness-Aware Model Evaluation*

- *Fairness Metrics:* Regularly assess models using metrics like demographic parity, equalized odds, and disparate impact.
- *Group vs. Individual Fairness:* Balance fairness at the group level (across demographics) and individual level (ensuring similar individuals are treated equally).

***3.5. Managing User Feedback and Avoiding Feedback Loops***

*3.5.1. Human-in-the-Loop Monitoring*

Instead of automatically adjusting model behavior based on user feedback, human-in-the-loop systems can be employed to review and validate feedback before making changes to the model. This reduces the risk of incorporating biased or incorrect feedback into the model. For example, in a language model for customer support, instead of allowing every user-corrected response to update the model, humans can review a sample of user feedback to ensure only accurate, unbiased corrections are used to improve the model.

*3.5.2. User-Specific Feedback Channels*

Rather than learning directly from all user interactions, models can benefit from selective feedback channels that separate useful corrections from biased or irrelevant data. These channels should be closely monitored to ensure that feedback is constructive. For example, in a product review sentiment analysis model, feedback from verified or frequent reviewers may be given more weight rather than learning from every rating given by users, as they are more likely to provide reliable feedback.

***3.6. Protecting Against Adversarial Attacks***

*3.6.1. Adversarial Training*

One of the most effective defenses against adversarial attacks is adversarial training, where the model is trained on examples that are intentionally designed to mislead it. This process strengthens the model's ability to resist attacks by

making it more robust to small perturbations in input data [24, 25]. The main types of adversarial training methodologies are highlighted below:

- *Basic Adversarial Training:* In basic adversarial training, adversarial examples are generated for each batch of data during training. These adversarial examples are combined with the original ones, allowing the model to learn both true and perturbed data distributions. This method enhances robustness against simple attacks. However, it may not defend against more sophisticated attacks like iterative or multi-step adversarial attacks.
- *Projected Gradient Descent (PGD) Adversarial Training:* PGD-based adversarial training is one of the most robust adversarial defense techniques. This approach uses multiple iterations of small gradient steps to generate adversarial examples. The model then trains on these stronger, multi-step adversarial examples, which are generally harder to defend against. This methodology significantly improves robustness against a wide range of adversarial attacks, as it exposes the model to high-complexity perturbations. However, PGD-based training is computationally intensive, as it requires generating multiple adversarial examples per input.
- *Adversarial Training with Regularization:* Techniques like TRADES (TRadeoff-inspired Adversarial DEfense via Surrogate-loss) incorporate a regularization term that balances the tradeoff between natural accuracy (on clean data) and adversarial robustness. This approach helps to optimize both robustness and generalization by regularizing the loss during training.
- *Domain-Specific Adversarial Training:* In domain-specific adversarial training, adversarial examples are crafted to represent domain-specific attack scenarios, such as specific perturbations in medical imaging or sensor noise in autonomous systems. The model is then trained on these specialized adversarial examples. This approach is highly effective in niche domains as it directly addresses common perturbation patterns within the domain. However, it may not generalize well to attacks outside the targeted scenarios.
- *Differential Privacy and Encryption*: Using techniques like differential privacy ensures that the model does not leak sensitive information even when exposed to
- adversarial examples. Additionally, encrypting the data pipeline reduces the risk of data manipulation.[26]
- *Defensive Distillation*: A defense mechanism that leverages *distillation*, a method traditionally used to transfer knowledge from a large model (teacher) to a smaller model (student) to resist adversarial perturbations. In this context, defensive distillation works by training the model in two stages:

1. *Distillation training:* A neural network is first trained normally (as a teacher model). Then, a second network (the student model) is trained to mimic the softened output probabilities (rather than hard labels) from the teacher model. A temperature parameter in the SoftMax function controls this softening, making the student model learn a more generalized representation of the data.

2. *Robustness against Adversarial Attacks*: The resulting student model tends to be less sensitive to small perturbations, which adversarial attacks rely on to mislead the model. The softened probabilities help the model focus on the overall structure of the data rather than fine-grained details, making it harder for adversarial examples to exploit weaknesses in the decision boundary. [27]

## 4. Conclusion

Deploying machine learning models into production presents various challenges, such as data drift, pipeline failures, limited interpretability, biased outputs, undesirable feedback loops, and adversarial attacks. To address these issues and ensure model reliability, machine learning engineers can adopt best practices like regular monitoring, comprehensive testing, promoting bias-free and interpretable models, implementing automated retraining, and employing adversarial training. As machine learning becomes increasingly integral to industry applications, establishing robust post-deployment strategies is crucial for maintaining these systems' long-term effectiveness and security.

## 5. Future Research and Open Challenges

Future research in ML monitoring and management is focused on addressing several key challenges, such as automating drift detection and adaptation, improving model interpretability and explainability, and ensuring the security and robustness of distributed and federated learning systems. The scalability of monitoring systems for large-scale deployments, privacy-preserving techniques for model evaluation, and continuous bias and fairness monitoring are also crucial areas for exploration. Additionally, there is a need for standardized frameworks for model transparency and auditing in regulated industries to ensure compliance with legal and ethical standards. Key open questions include efficiently detecting drift, balancing privacy with model performance, and mitigating bias in real-time deployments. By addressing these challenges, future research will enable more robust, secure, fair, and scalable ML systems, ensuring their safe and effective use across a wide range of applications.

## Conflicts of Interest

The authors receive no financial compensation for this work, and the views expressed are solely their own, not those of their employer. The authors declare that there is no conflict of interest regarding the publication of this paper.

## Funding Statement

The authors declare that this research was self-funded and part of an independent research

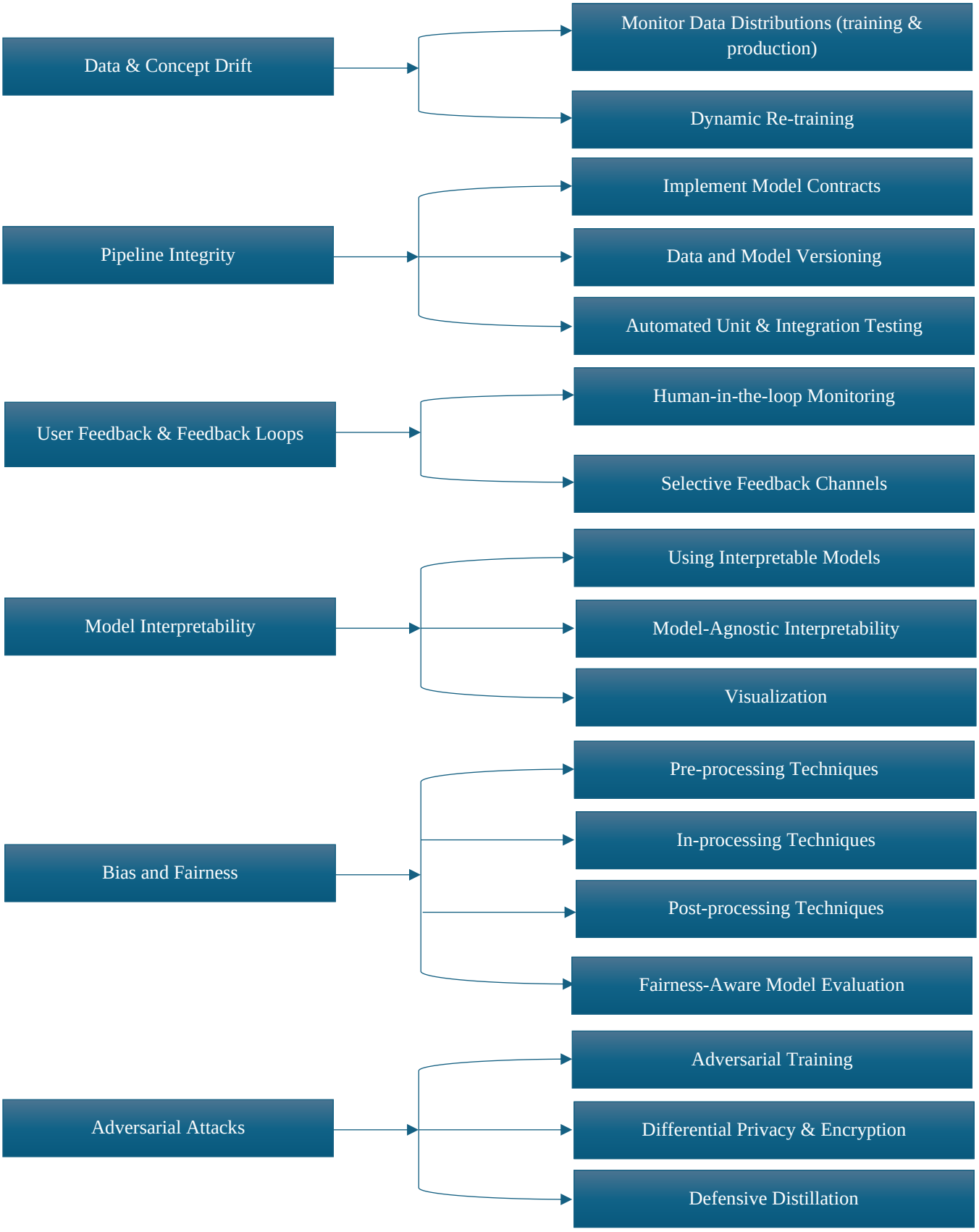


**Fig. 2 Challenges and Key solutions**

**Table 1. Common tools and frameworks for real-time monitoring of ML models**

| Tool/Framework | Description |
|---|---|
| Prometheus & Grafana [28] | Open-source monitoring tools used to track system and model performance metrics. Prometheus collects time-series data, while Grafana visualizes it in customizable dashboards. Best suited for large-scale systems but lacks specific features for ML models. |
| Evidently AI [29] | A specialized tool for real-time monitoring of ML models, focusing on data drift, performance tracking, and model evaluation. Offers pre-built reports for easy interpretation but may require customization for advanced use cases. |
| Arize AI [30] | A dedicated platform for ML observability and troubleshooting, offering real-time monitoring of model performance, drift detection, and error analysis. Provides powerful visualizations and alerts, but is proprietary and may require integration work. |
| Fiddler AI [31] | Specializes in explainable AI (XAI) and model transparency, providing tracking for model performance, fairness, and interpretability. Customizable metrics and alerts are available, though it may be costly for large deployments. |
| Amazon SageMaker Model Monitor [32] | A built-in tool within AWS SageMaker for monitoring models deployed on the AWS cloud. It provides drift detection and performance tracking, making it ideal for teams already using AWS services but limited to AWS environments. |
| Azure Monitor & MLOps [33] | Native to the Azure ecosystem, it integrates directly with Azure Machine Learning for monitoring logs, model metrics, and alerts. It is scalable and cloud-native but less flexible for multi-cloud setups. |

## References


[1] Jie Lu et al., "Learning under Concept Drift: A Review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346-2363, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Joao Gama et al., "A Survey on Concept Drift Adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1-37, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[3] Sungsoo Ray Hong et al., "Human Factors in Model Interpretability: Industry Practices, Challenges, and Needs," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, pp. 1-26, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Harini Suresh, and John V. Guttag, "A Framework for Understanding Unintended Consequences of Machine Learning Life Cycle," *EAAMO '21: Proceedings of the 1st ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, Newyork, USA, pp. 1-9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] David Carless, "Feedback Loops and the Longer-Term: Towards Feedback Spirals," *Assessment & Evaluation in Higher Education*, vol. 44, no. 5, pp. 705-714, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Nicolò Pagan et al., "A Classification of Feedback Loops and their Relation to Biases in Automated Decision-Making Systems," *EAAMO '23: Proceedings of the 3rd ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, Boston MA USA, pp. 1-14, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[7] Rohan Taori, and Tatsunori Hashimoto, "Data Feedback Loops: Model-Driven Amplification of Dataset Biases," *Proceedings of the 40th International Conference on Machine Learning*, pp. 33883-33920, 2023. [Google Scholar] [Publisher Link]

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and Harnessing Adversarial Examples," *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1-11, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[9] Anirban Chakraborty et al., "A Survey on Adversarial Attacks and Defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25-45, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Battista Biggio et al., "Evasion Attacks Against Machine Learning at Test Time," *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013*, pp. 387-402, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[11] Zhiyi Tian et al., "A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1-35, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures," *CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver Colorado USA, pp. 1322-1333, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[13] Andika Rachman, Tieling Zhang, and R.M. Chandima Ratnayake, "Applications of Machine Learning in Pipeline Integrity Management: A State-of-the-Art Review," *International Journal of Pressure Vessels and Piping*, vol. 193, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, "Model-Agnostic Interpretability of Machine Learning," *arXiv*, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[15] Edoardo Mosca et al., "SHAP-Based Explanation Methods: A Review for NLP Interpretability," *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, pp. 4593-4603, 2022. [Google Scholar] [Publisher Link]

[16] Qinglong Zhang, Lu Rao, and Yubin Yang, "A Novel Visual Interpretability for Deep Neural Networks by Optimizing Activation Maps with Perturbation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, pp. 3377-3384, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Ahmed Alqaraawi et al., "Evaluating Saliency Map Explanations for Convolutional Neural Networks: A User Study," *IUI '20: Proceedings of the 25th International Conference on Intelligent User Interfaces*, Cagliari Italy, pp. 275-285, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Julia Moosbauer et al., "Explaining Hyperparameter Optimization via Partial Dependence Plots," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2280-2291, 2021. [Google Scholar] [Publisher Link]

[19] Bobby Yan, Skyler Seto, and Nicholas Apostoloff, "FORML: Learning to Reweight Data for Fairness," *arXiv*, pp. 1-9, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Shubham Sharma et al., "Data Augmentation for Discrimination Prevention and Bias Disambiguation," *AIES '20: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Newyork, United States, pp. 358-364, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[21] Ehsan Adeli et al., "Representation Learning with Statistical Independence to Mitigate Bias," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, *Conference on Applications of Computer Vision (WACV)*, pp. 2513-2523, 2021. [Google Scholar] [Publisher Link]

[22] Jongin Lim et al., "Bias-Adv: Bias-Adversarial Augmentation for Model Debiasing," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR)*, pp. 3832-3841, 2023. [Google Scholar] [Publisher Link]

[23] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma, "Fairness-Aware Learning through Regularization Approach," *2011 IEEE 11th International Conference on Data Mining Workshops*, Vancouver, BC, Canada, pp. 643-650, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[24] Tao Bai et al., "Recent Advances in Adversarial Training for Adversarial Robustness," *arXiv*, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ali Shafahi et al., "Adversarial Training for Free!," *Advances in Neural Information Processing Systems* 32 *(NeurIPS 2019)*, 2019. [Google Scholar] [Publisher Link]

[26] Trung Ha et al., "Differential Privacy in Deep Learning: An Overview," *2019 International Conference on Advanced Computing and Applications (ACOMP)*, Nha Trang, Vietnam, pp. 97-102, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[27] Nicolas Papernot et al., "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks," *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, pp. 582-597, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[28] Get started with Grafana and Prometheus, Grafana Labs. [Online]. Available: https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/

[29] Evidently AI, Collaborative AI Observability platform, Evaluate, Test, and Monitor your AI-Powered Products. [Online]. Available: https://www.evidentlyai.com/

[30] Arize AI, AI Observability and Evaluation Platform. [Online]. Available: https://arize.com/

[31] Fiddler AI, Enterprise AI Observability. [Online]. Available: https://www.fiddler.ai/

[32] Data and Model Quality Monitoring with Amazon SageMaker Model Monitor. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html

[33] Machine Learning Operations (MLOps), Azure Monitor & MLOps. [Online]. Available: https://azure.microsoft.com/en-us/solutions/machine-learning-ops