MathWorks® Accelerating the pace of engineering and science

# Model Monitoring and Drift Detection: Ensuring the Health and Fairness of Deployed Models

The paper focuses on maintaining the integrity and fairness of machine learning models as they are increasingly used in decision-making. It emphasizes the importance of implementing fairness metrics and drift detection mechanisms to ensure models remain unbiased and accurate over time. The paper discusses the role of global regulations in promoting responsible AI, the utility of dashboards for real-time model monitoring, and the necessity of automated alerts for timely interventions. It also highlights the significance of a continuous, proactive approach to model risk management (MRM), which includes the integration of tools like Modelscape for comprehensive oversight and the operationalization of model performance, relevance, and fairness throughout the model lifecycle.

## Model Monitoring: A Necessity, not a Choice

Effective model monitoring is essential for the success of machine learning (ML) models, especially as they face rapidly changing environments and adaptive challenges. The dynamic nature of ML models requires constant vigilance to ensure they remain accurate and effective. Regulatory expectations, such as those outlined in the PRA's SS3/18 paper, already emphasize the importance of model monitoring. The COVID-19 pandemic highlighted the need for this, as many models based on pre-pandemic assumptions required adjustments. Using model overlays, where expert judgment is applied to modify outputs, is a common mitigation strategy. However, uncoordinated changes across interconnected models can lead to failures. Therefore, ongoing, and timely adjustments are crucial, not just periodic revalidations, to maintain the integrity and performance of ML models in the face of evolving conditions.

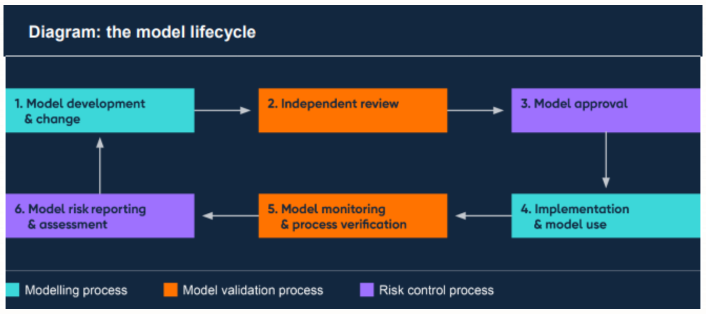


Diagram: the model lifecycle
1. Model development & change
2. Independent review
3. Model approval
4. Implementation & model use
5. Model monitoring & process verification
6. Model risk reporting & assessment
Modelling process
Model validation process
Risk control process

*Figure 1: The model lifecycle as mentioned in* ***SS/123***

## Advancing Model Integrity with Fairness Metrics and Drift Detection

As machine learning models become integral to decision-making, maintaining their integrity through fairness metrics and drift detection is crucial. These measures are increasingly recognized in global regulations, such as the Monetary Authority of Singapore's FEAT principles (Fairness, Ethics, Accountability, Transparency) and the European AI Act. Additionally, the Federal Reserve in the United States is paying closer attention to these aspects in AI and ML models. An EY ***white paper*** underscores the concept of responsible AI, positioning it at the heart of trustworthy AI systems, surrounded by various critical factors, with today's focus being specifically on model performance.

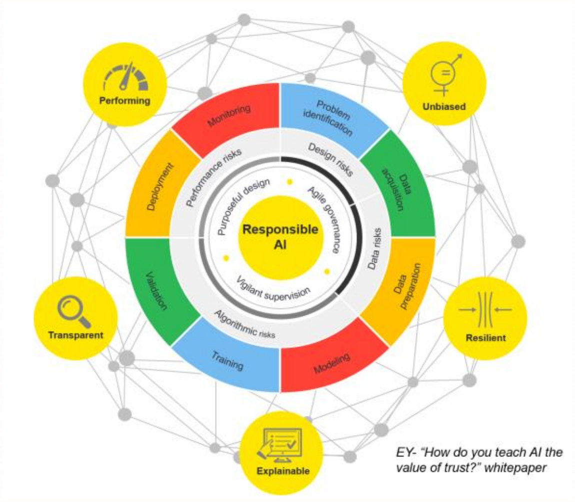


*Figure 2: EY's Five Attributes of Responsible AI*

## Dashboards enable Dynamic MRM

Dashboards have become a pivotal tool in managing model risk by providing real-time insights and acting as an early warning system for model performance issues. They enable the detection of data drift and help ensure fairness, confirming that models operate without bias. They offer a centralized view of performance and risks and help organizations coordinate responses to issues that may affect not just a single model but a suite of interconnected models, thereby mitigating the risk of uncoordinated actions that could impact model integrity and business operations.

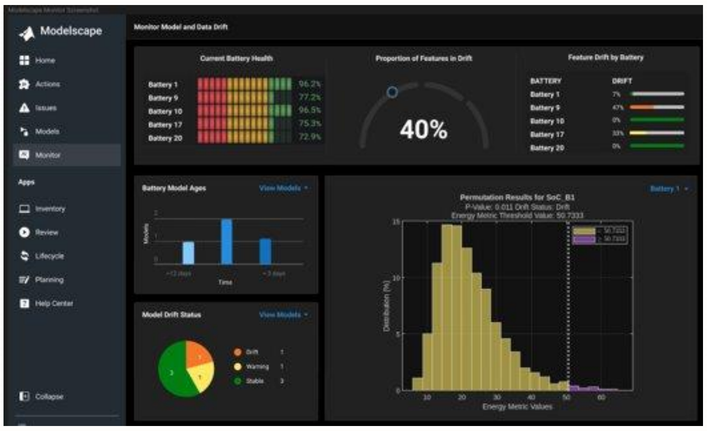


*Figure 3: A Centralized View Dashboard of Performance and Risks*

## Timely Alerts Mitigate Risk by Prompting Immediate Action

Automated alerts are crucial for timely interventions in model monitoring, ensuring that any deviations or issues are addressed promptly within the governance and development processes. Embedding these alerts into model risk management allows for a proactive approach to maintaining model integrity. Dashboards are useful tools for monitoring, but their effectiveness is contingent on active observation. Automated alert systems are necessary to notify relevant stakeholders for immediate action. These systems should facilitate not just the production but also the consumption of alerts, such as those indicating model bias or data drift. The incorporation of a comprehensive observability framework, as illustrated by the integration of microservices, Kubernetes, and cloud infrastructure with an Open Telemetry Collector in the system architecture, enhances the precision and scope of such alerts (Figure 4). The alerts can trigger various responses, like recalibration or redevelopment of models. This approach completes the model risk management lifecycle, ensuring models remain accurate and reliable. This overview emphasizes the shift from viewing model monitoring as a periodic check to recognizing it as a continuous, integral process that drives improvements through timely alerts.

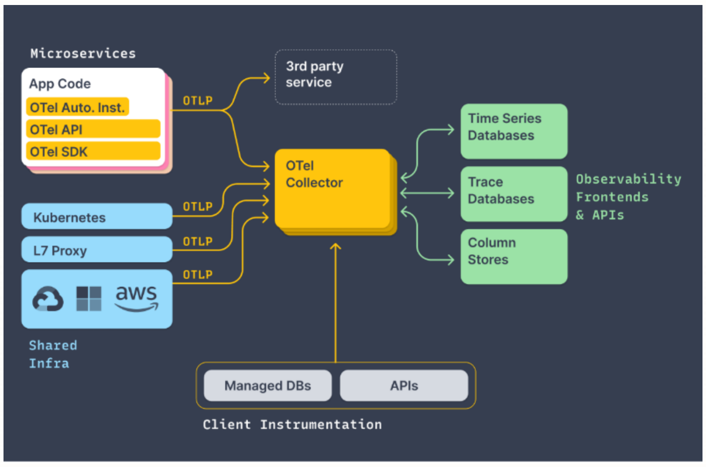
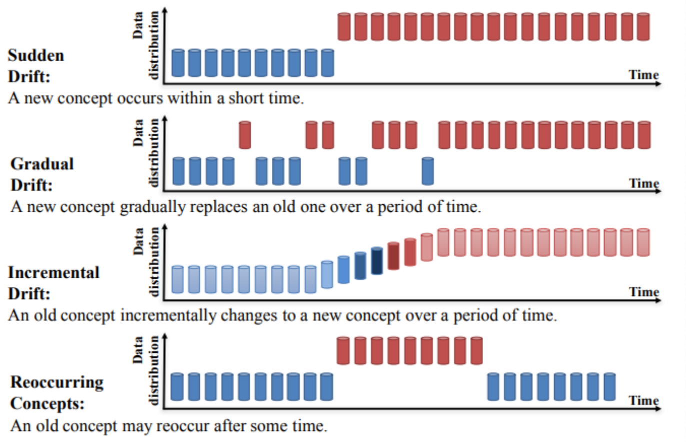


*Figure 4: Cloud Observability with Open Telemetry*

## Enhancing Trust in Models through Drift and Fairness Monitoring

Drift refers to the changes in data patterns over time, which can be sudden or gradual (Figure 5), affecting the model's performance. It is essential to detect and adjust for these changes to ensure the model's current relevance. Fairness, on the other hand, involves assessing the model's decisions across different groups, potentially divided by protected attributes like gender or race, to ensure unbiased outcomes. By actively observing and addressing both drift and fairness, one can significantly improve the integrity and fairness of their models, thereby fostering greater trust in their applications.



***Source**: **https://arxiv.org/pdf/2004.05785.pdf***

*Figure 5: An Example of concept drift types.*

## Example: NYC House Price Data

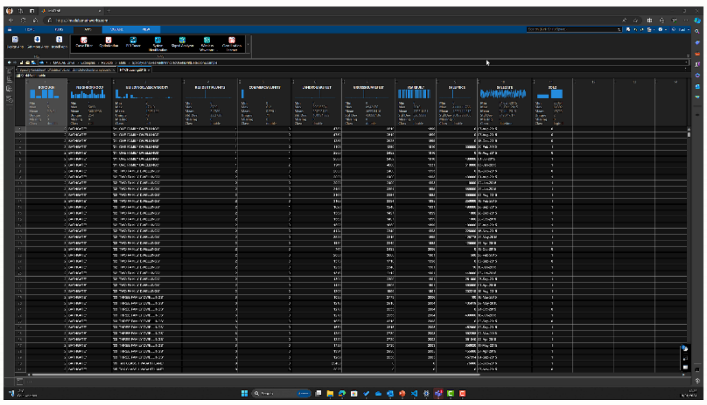


*Figure 6: NYC House Prices 2015*

The New York City House price dataset for 2015 presents a classic structure where each row represents an observation and each column stands for a variable, with a mix of categorical and numeric types. Notably, the 'sales date' column introduces a temporal element, marking when each observation occurred. In building a machine learning model, it is common to treat each row individually and initially disregard the 'sales date', which can be a reasonable starting approach. However, if the variables' distribution shifts over time, and 'sales date' is the only temporal indicator, the model may be prone to drift. This highlights the importance of considering temporal changes, especially when predicting binary outcomes like whether a property was sold, to ensure the model remains robust over time.

## Continuous Monitoring Protects Against Model Decay

We will start by training the model, followed by careful observation to identify any issues that arise during its operation. Once problems are detected, the final step is to implement corrective actions to rectify the performance issues.

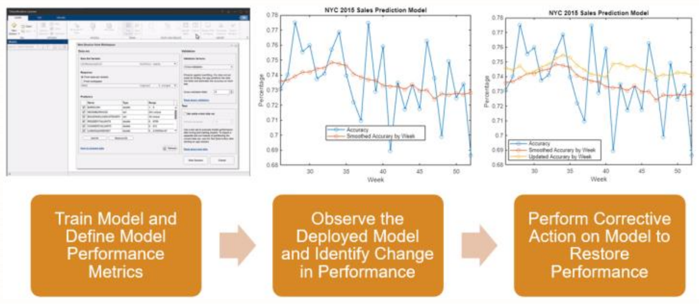

NYC 2015 Sales Prediction Model
Accuracy
Smoothed Accuracy by Week
Updated Accuracy by Week
Percentage
Week
Train Model and Define Model Performance Metrics
Observe the Deployed Model and Identify Change in Performance
Perform Corrective Action on Model to Restore Performance

*Figure 7: Data Drift Three Steps Forumula*

***The Classification Learner App*** is a great asset when it comes to model training within MATLAB. After loading our model into the app, it meticulously processes the datasets, applying rigorous best-in-practice techniques like cross-validation, which are instrumental in mitigating the risk of overfitting. This ensures that the model's predictive power is genuine and not just an artifact of the noise within the training data. While we work under the assumption that our model is stationary, the Classification Learner app aids us in efficiently building a simple decision tree. This decision tree, though fundamental in structure, is optimized for performance by the app's algorithms and user-friendly interface, which simplifies the complexities of machine learning workflows. The app's value lies in its ability to not only develop models but also to understand and improve them interactively.

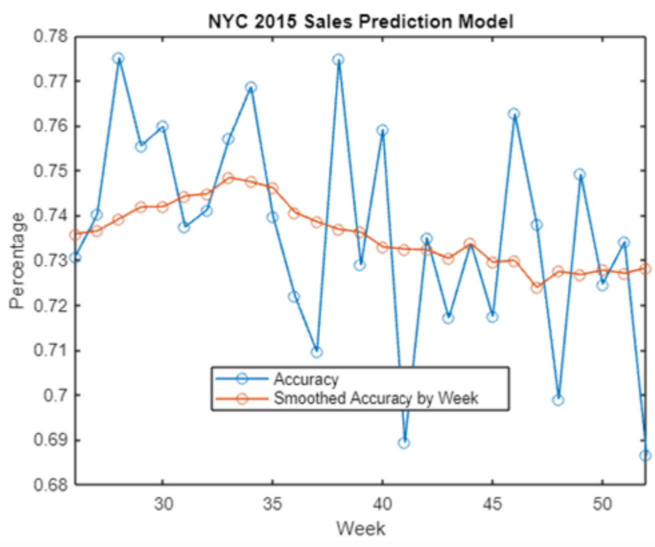


*Figure 8: NYC Sales Prediction Model*

The analysis involves assessing the accuracy of a model in predicting house sales on specific dates, acknowledging that the housing market is unpredictable with a noisy signal but exhibits a clear trend. The model's accuracy, initially around 73%, appears to decline over time. To address this, a strategy called incremental machine learning is suggested, where new data is periodically added to the model to maintain its relevance. Additionally, weighting can be applied to give more importance to recent data, as part of a strategy to adapt the model to changing market conditions.

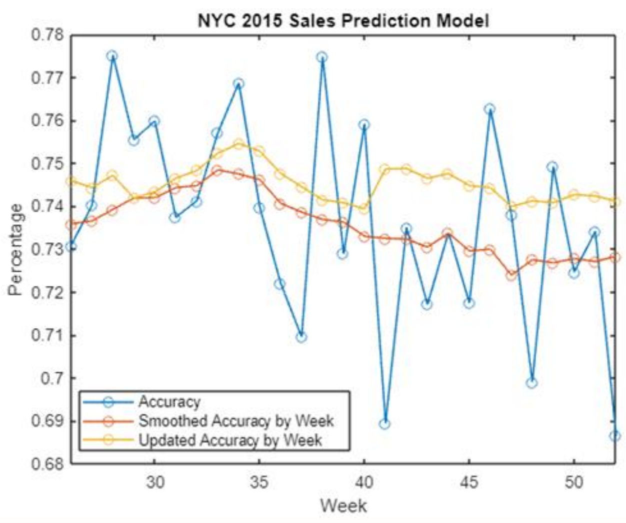


*Figure 9: NYC Sales Prediction Model with Updated Weekly Accuracy*

An updated model was applied to the same dataset after the original model experienced significant drift and exceeded acceptable thresholds. The model was retrained, leading to an immediate improvement in accuracy, which was maintained over time. Corrective actions taken in Week 40 ensured that the model's performance by the end of the year remained stable and less prone to data drift, sustaining its original performance level.

In Summary, the workflow discussed involves a three-step process for managing a binary classification model:

1. Determine the metric to measure, which in this case is the accuracy of the model's predictions.

2. Establish a threshold for action; if the model's accuracy falls below 73%, it triggers a retraining of the model.

3. Once the model is retrained due to a significant drop in accuracy, it is redeployed, resulting in improved performance.

## Instrumentation and Alerting in Model Production

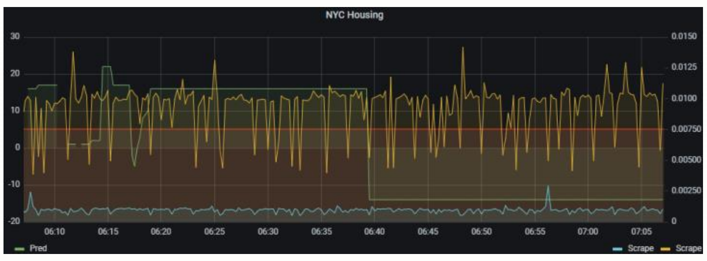


*Figure 10: NYC Housing Predictive Analysis and Real-Time Data Fluctuations*

To ensure models perform well in production, they are instrumented with metrics that can alert developers to issues such as data drift. Using tools like MATLAB, models can be set up to send metrics to databases, allowing for real-time monitoring. For instance, if data drift is detected, the model can be adjusted to smooth out the noisy signals and set thresholds to trigger alerts. Modelscape, a Model Risk Management solution, supports various databases and allows developers to experiment with alerting systems before deployment. Alerts can be configured to notify the appropriate teams, and strategies for handling multiple or false alerts include grouping similar alerts and suppressing or silencing known issues. It's also important to consider external factors affecting model performance, such as system utilization and execution delays, which underscores the need for a comprehensive approach to model monitoring that extends beyond the development phase.

## Integrating DevOps in AI/ML Model Monitoring

When analyzing time series data for spikes that may impact model performance, it's essential not only to consider model-specific metrics, such as drift and fairness but also the operational aspects like execution time and data throughput. This requires a comprehensive DevOps infrastructure that continuously monitors and measures model properties over time. Model developers are tasked with creating the necessary instrumentation within their models and collaborating with DevOps teams to ensure these models are integrated into a larger monitoring platform. Tools like Prometheus can be used to collect telemetry data, allowing for real-time queries about model performance. Companies like MathWorks offer solutions like Modelscape to facilitate this process. Ultimately, the observability of models—a concept borrowed from DevOps—combines both instrumentation and telemetry, enabling DevOps teams to monitor models effectively and provide valuable feedback to business users and developers, aligning with model governance and development protocols.

## Modelscape Solution

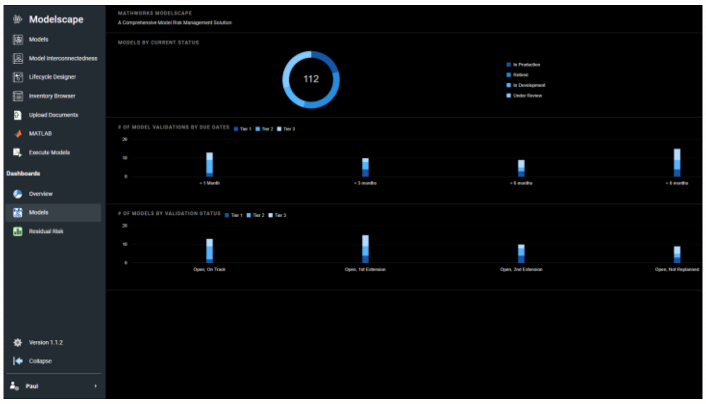


Figure 11: MathWorks Modelscape Dashboard

Modelscape allows for an organization-wide overview of all models, enabling the creation of dashboards for high-level insights and the ability to delve into specific models for detailed analysis. The system integrates alerts from the monitoring system, flagging issues for models pending approval and for those in operation. Each alert can be assigned a creation date and severity level, either automatically or through manual review, and tracked within the model governance process. This process determines whether a model should be adjusted or completely redeveloped. Model monitoring thus serves as a crucial feedback mechanism, informing governance and development to enhance model performance, relevance, and fairness.

## Finally: Ensure the Health and Fairness of Deployed Models using Modelscape

MathWorks has developed Modelscape, a comprehensive solution that tackles the challenges of model fairness and data drift. Modelscape Monitor offers out-of-the-box features such as threshold setting, alerts, and dashboard creation for performance review. It supports model refinement based on production data and informs governance decisions, like whether to update or retire models. Additionally, Modelscape aids the entire modeling lifecycle, including validation in live environments, preproduction testing, and deployment. This holistic monitoring strategy enhances risk management and operational efficiency.

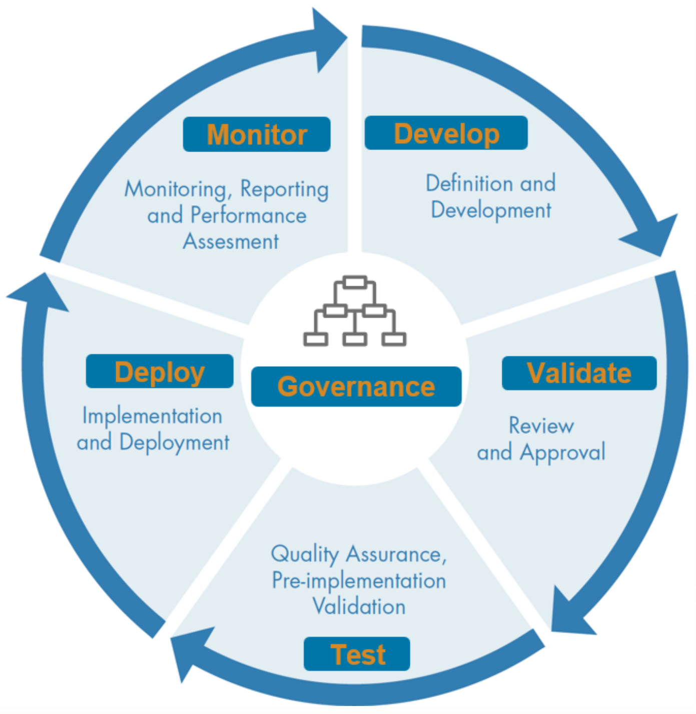

Monitor
Monitoring, Reporting and Performance Assesment
Develop
Definition and Development
Governance
Deploy
Implementation and Deployment
Validate
Review and Approval
Quality Assurance, Pre-implementation Validation
Test

Figure 12: Model Risk Management Lifecycle

*» Learn more: mathworks.com/modelscape*