Towards Runtime Monitoring for Responsible Machine Learning using Model-driven Engineering

Hira Naveed
Monash University
Australia
hira.naveed@monash.edu

John Grundy
Monash University
Australia
john.grundy@monash.edu

Chetan Arora
Monash University
Australia
chetan.arora@monash.edu

Hourieh Khalajzadeh
Deakin University
Australia
hkhalajzadeh@deakin.edu.au

Omar Haggag
Monash University
Australia
omar.haggag@monash.edu

ABSTRACT

Machine learning (ML) components are used heavily in many current software systems, but developing them responsibly in practice remains challenging. *'Responsible ML'* refers to developing, deploying and maintaining ML-based systems that adhere to human-centric requirements, such as fairness, privacy, transparency, safety, accessibility, and human values. Meeting these requirements is essential for maintaining public trust and ensuring the success of ML-based systems. However, as changes are likely in production environments and requirements often evolve, design-time quality assurance practices are insufficient to ensure such systems' responsible behavior. Runtime monitoring approaches for ML-based systems can potentially offer valuable solutions to address this problem. Many currently available ML monitoring solutions overlook human-centric requirements due to a lack of awareness and tool support, the complexity of monitoring human-centric requirements, and the effort required to develop and manage monitors for changing requirements. We believe that many of these challenges can be addressed by model-driven engineering. In this new ideas paper, we present an initial meta-model, model-driven approach, and proof of concept prototype for runtime monitoring of human-centric requirements violations, thereby ensuring responsible ML behavior. We discuss our prototype, current limitations and propose some directions for future work.

CCS CONCEPTS

• **Software and its engineering** → **Requirements analysis**; **Software safety**; **Maintaining software**; *Risk management*; **Operational analysis**; **Software defect analysis**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Runtime monitoring, Responsible ML, Human-centric requirements, Machine learning components, Model-driven engineering

MODELS '24, September 22–27, 2024, Linz, Austria

ACM ISBN 979-8-4007-0504-5/24/09
https://doi.org/10.1145/3640310.3674092

ACM Reference Format:
Hira Naveed, John Grundy, Chetan Arora, Hourieh Khalajzadeh, and Omar Haggag. 2024. Towards Runtime Monitoring for Responsible Machine Learning using Model-driven Engineering. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3640310.3674092

1 INTRODUCTION

Over the last decade, machine learning (ML) components have emerged and become prevalent in software systems to support use cases such as predictive analysis, sales forecasting, fraud detection, medical diagnosis, and much more [59, 64]. While the advent of ML-based systems has created many benefits and opportunities, it has also introduced several new software engineering (SE) and modelling challenges. One of these challenges is preventing the misapplication, misuse, or potential harm caused by ML-based systems [31]. This challenge can be addressed by developing, deploying, and maintaining ML-based systems *responsibly*, i.e., *responsible ML* [19, 51]. Responsible ML entails that ML-based systems conform to critical human-centric requirements such as fairness, privacy, safety, trust, transparency, accessibility, sustainability and well-being [19, 51]. The lack of responsible ML practices results in ML-based systems that are biased, privacy-invasive, unsafe, and untrustworthy, thus violating several human-centric requirements [12, 66]. The likelihood of such violations in ML-based systems is evident, as even large companies have experienced them, e.g., Amazon's gender-biased recruitment tool [23], IBM's unreliable cancer diagnosis [69], and Microsoft's offensive Twitter chatbot [17].

Given the inherent uncertainty of ML-based systems, relying solely on responsible ML practices for development and testing is not sufficient – many human-centric requirements can be violated after the system has been deployed [75]. Unforeseen changes in incoming data, changes in operating context, and anomalies can cause the ML-based system to behave unexpectedly [22, 29]. For example, when the COVID-19 pandemic hit, existing ML models experienced a significant drop in predictive performance due to changes in the operating context [60]. Such changes can impact fairness metrics, potentially harming sensitive groups [29, 49] and weakening defenses against privacy attacks [54]. Additionally, unintentional mistakes, such as bugs in the ML pipeline can cause issues after deployment.

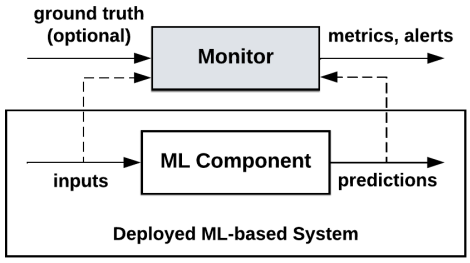

Figure 1: Runtime Monitoring of ML Systems

Runtime monitoring of ML-based systems for human-centric requirements violations is recommended as an essential part of responsible ML deployment and maintenance [75]. Runtime monitoring refers to continuously observing the behavior of a deployed ML-based system by capturing and analyzing relevant data to identify unwanted behavior that violates system requirements [56]. A basic architecture of how runtime monitoring for ML-based systems is conducted is shown in Figure 1. The monitor collects inputs and/or predictions of the ML component, computes pre-defined metrics of interest, and alerts ML engineers if unwanted behavior is detected. Some examples of frequently encountered unwanted behavior in ML components are performance drops, data drift (changes in data patterns), concept drift (changes in operating context), prediction drift (changes in prediction patterns), and anomalies in inputs. Optionally, the monitor may receive ground truth data, which can be available with a short delay (e.g., button clicks) or a long delay (e.g., loan repayments).

At present, numerous approaches and tools exist for runtime monitoring of ML-based systems, most focusing entirely on technical aspects such as performance degradation and drift [30, 32, 48, 65, 73]. However, these approaches often fail to monitor human-centric requirements violations and the consequences of technical requirements violations on human-centric requirements [56]. The absence of such monitoring raises several problems: 1) delayed detection of human-centric requirements violations, relying on user feedback or tedious manual observations; 2) frustration among users and reduced trust in the ML-based system; 3) reputational damage, financial losses, and legal consequences for businesses. Therefore, a key step of responsible ML is to ***monitor ML-based systems for human-centric requirements violations*** [56, 75]. In practice, monitoring ML-based systems for human-centric requirements violations is far from trivial. Deploying and maintaining ML-based systems, in general, is challenging due to a lack of technical skills in the market [5], lack of awareness and understanding regarding responsible ML among practitioners [53], or the fact that most organizations find it difficult to monitor ML models at runtime [55].

Implementing runtime monitors for responsible ML involves several activities. ML engineers must implement monitors for various human-centric requirements of interest and integrate them with the ML infrastructure. The lack of ground truth data and subjectivity of some human-centric requirements such as pleasure, trust, and autonomy make direct monitoring difficult and proxy measures need to be specified [35]. Furthermore, as requirements evolve, the monitors must be updated and maintained to capture the current properties of interest, this is also essential for efficient resource management [16, 52]. Maintaining the monitors for human-centric and technical requirement violations requires significant time and effort from ML engineers, as they must ensure consistency across all monitors [43]. These monitors may be built from scratch or use existing toolkits, further exacerbating the maintenance issue. While monitoring is the responsibility of the ML engineer, organizations may not always have a designated role for this task. Depending on the organization, the monitoring process may be conducted by multidisciplinary teams with employees in various other roles (e.g., data scientists and software engineers); this becomes challenging as all have different expertise and preferences [55].

We believe that model-driven engineering (MDE) can address many of these challenges. The abstraction of models can hide many complexities and improve collaboration, whereas automated model transformations can reduce development and maintenance effort for runtime monitors [72]. MDE has been successfully applied for runtime monitoring of traditional software systems [47, 72] and recently for performance monitoring of ML-based systems [43, 44]. However, its application to runtime monitoring for human-centric requirement violations in ML-based systems, crucial for ensuring responsible behavior, remains unexplored. As runtime monitoring for responsible ML is challenging and often overlooked, we address this gap by presenting an initial approach to make it easier and faster. In this new ideas paper, we present an initial meta-model of our domain-specific language (DSL), *MoReML* – Monitoring Responsible ML, an example of a model created in MoReML, a model-driven approach, and a proof of concept prototype[1] for runtime monitoring of human-centric requirements violations in ML components, thereby helping to ensure more responsible ML. We also incorporate performance and drift monitors, as they may impact the violations of human-centric requirements. We discuss how models can be created in MoReML, how runtime monitors can be generated from these models, the current limitations in our work, and future directions.

2 RUNTIME MONITORING FOR RESPONSIBLE ML

Our research is based on the following research questions:

- **RQ1.** Does the DSL (MoReML) effectively capture and represent ML monitoring aspects and human-centric requirements in a comprehensible manner for engineers to conduct runtime monitoring?
- **RQ2.** How well can the generated runtime monitors detect violations of specified human-centric requirements (e.g., fairness, privacy) and technical requirements (e.g. performance, drift)?

2.1 Our Approach

Figure 2 provides an overview of our proposed approach. (1) The ML expert identifies the key human-centric requirements of interest, which may vary depending on the context, goal, and application domain of the ML-based system. For example, fairness (concerning race or gender) is an important human-centric requirement in credit score prediction systems. (2) The ML expert specifies technical requirements corresponding to the high-level human-centric

[1] Prototype available at https://zenodo.org/records/11394260

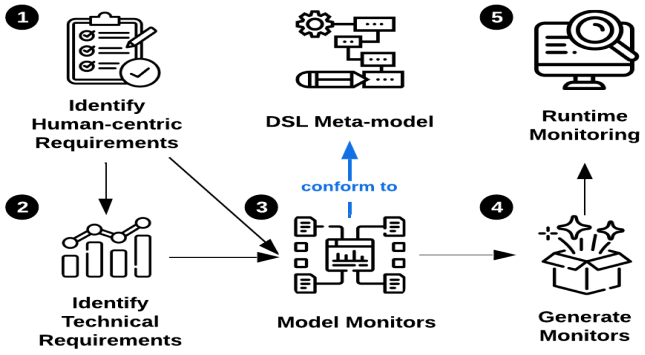

Figure 2: Our Proposed Approach

requirements, such as fairness metrics and thresholds. (3) The ML expert constructs a model describing the required monitors. To ensure this model conforms to the MoReML meta-model, we specify constraints using the Epsilon Validation Language (EVL) [42]. The model includes the ML component, the monitoring system, monitors of interest (e.g., fairness monitor), and the linked requirements. (4) Our tool generates code for the runtime monitors by applying model-driven engineering. (5) The generated monitors monitor the ML component at runtime. The monitors collect data from the ML component and compute metrics to identify violations of the specified requirements.

2.2 Monitoring Meta-Model

Models provide several benefits, including reduced complexity, easier stakeholder communication, automated code generation, and increased productivity [33, 74]. While general-purpose modeling languages such as UML [26] are generic and complex, a domain-specific language (DSL) is relatively much easier to use as it captures domain-specific information in familiar terminology [28, 33]. The abstract syntax for a DSL is specified through a meta-model and the concrete syntax may be textual, visual, or both. In this work, we present our initial effort towards a DSL, *MoReML*, to capture runtime monitoring concepts for responsible ML behavior.

We present an initial meta-model of MoReML that enables the specification of the monitoring system, the ML component being monitored, runtime monitor characteristics, key runtime monitoring metrics, and the linked high-level human-centric requirements. The concepts in the meta-model are based on the findings from our systematic literature review [57]. Figure 3 shows the classes and their relationships in our initial meta-model. The rationale and purpose of each key class in the meta-model are explained below.

2.2.1 MonitoringSystem. This is the top-most class that represents the entire ML monitoring system. It is essential for controlling all monitors, managing datasets, and interacting with the ML component. The *MonitoringSystem* class has the attributes name and a unique id. It is linked to exactly one *MLcomponent* being monitored and contains one or more *Dataset* and *Monitor* class instances.

2.2.2 MLcomponent. This class describes the ML component being monitored. Specifying the ML component allows tailored monitoring suitable to the component characteristics such as the ML algorithm. An ML component is similar to a traditional software component with code and input/output interfaces. The ML component contains code of exactly one trained ML model and interfaces for inputs (ML model inputs) and outputs (ML model predictions). This class also contains the model ID, and ML algorithm (e.g. Logistic Regression).

2.2.3 Dataset. This class represents the datasets available for monitoring and baseline comparison, these can be training, testing, or production datasets. The dataset class is needed for specifying datasets used by the ML component, tracking changes, and . We assume that monitoring is being done in batches with the production dataset. The other alternative would be to monitor each in real-time. For easier version control and analysis, the *MonitoringSystem* maintains a record of all the *Dataset* objects. The *Dataset* class consists of attributes for dataset path, label name (prediction column name), column names and categorical features needed to load the dataset.

2.2.4 Monitor. This class represents a runtime monitor that is a part of the monitoring system. We include this in the meta-model to represent a generic monitor, its characteristics, and its relations to other concepts. A monitor offers ML engineers the flexibility to activate or deactivate it based on the current requirements of interest and a defined interval for computing metrics periodically. These capabilities enable efficient resource management and the adaptability of monitors. Each monitor contains one instance of *MonRequirement* class representing the high-level requirement being monitored and one instance of the *Alert* class representing the alert triggered if the requirement is violated. The low-level technical requirements linked to these high-level requirements are specified through the metric classes.

The *Monitor* is a generic class further specified into the four types of monitors we consider in this work i.e., *FairnessMonitor*, *PrivacyMonitor*, *PerformanceMonitor*, and *DriftMonitor*. Each monitor type has different attributes and metrics so we define them as separate classes in the meta-model. Monitoring human-centric requirements in isolation is not as effective as monitoring them in combination with more technical aspects such as performance and drift. A single holistic monitoring solution provides a unified view of the ML-based system's behavior and eliminates the complexity of managing multiple monitoring tools.

The *FairnessMonitor* class describes a monitor that receives a dataset containing model predictions and another dataset including ground truth labels. It computes fairness metrics for the specified protected attributes (e.g., sex, race) and privileged groups (e.g., Male, Caucasian) to assess whether the ML model behaves fairly or not. The *FairnessMonitor* class contains one or more instances of the *FairnessMetric* class which has an enumeration attribute name for metric type (e.g., statistical parity difference) and a threshold.

Similarly, the *PerformanceMonitor* class refers to a monitor that analyzes a dataset with model predictions and another dataset with ground truth labels to calculate performance metrics (e.g., accuracy). The *PerformanceMonitor* class contains one or more instances of the *PerformMetrics* classes that represent the performance metrics. This monitor is useful for detecting any performance drops in the ML model.

The *DriftMonitor* class describes a monitor that identifies changes in data patterns between the production and training datasets. It

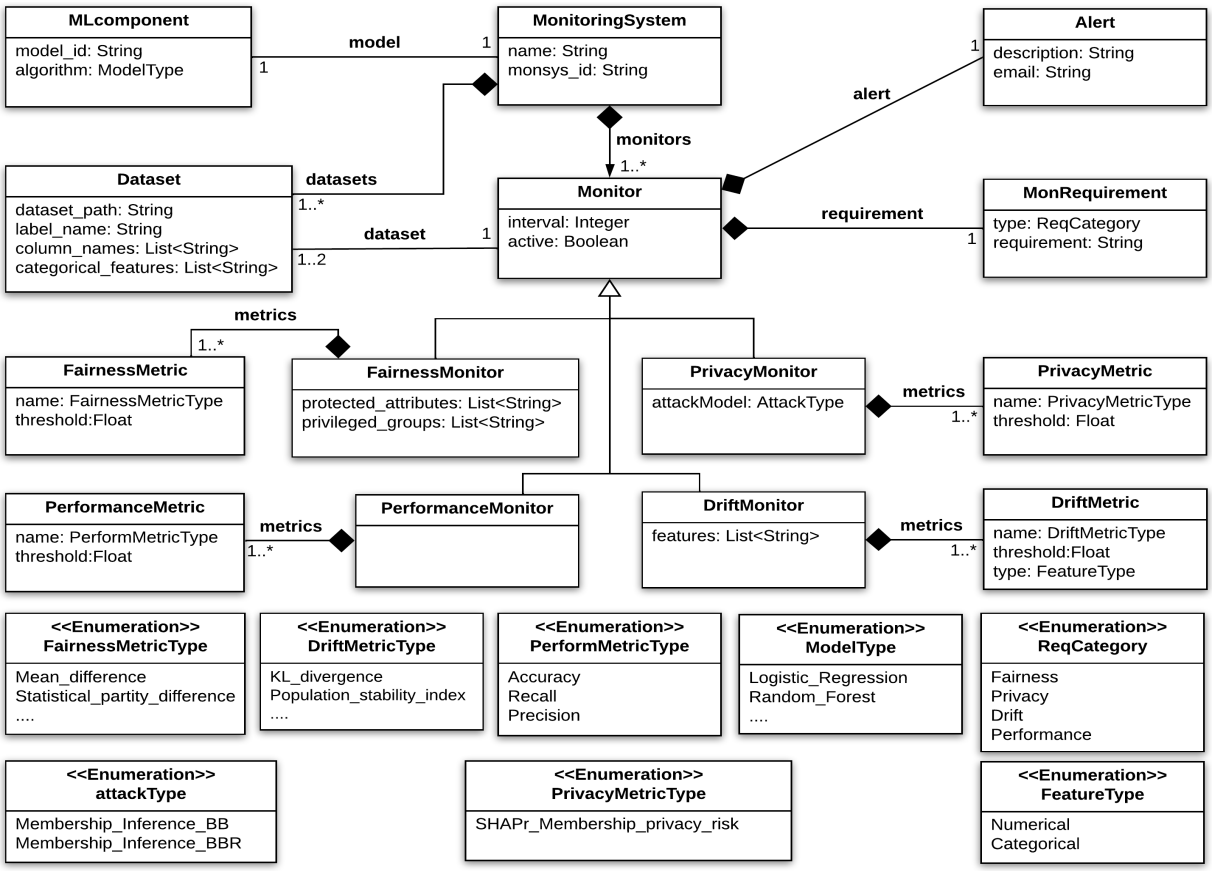

Figure 3: Meta-model of MoReML

has a feature attribute to specify the input data features under observation for drift detection. Various drift metrics (e.g., Kolmogorov–Smirnov) can be used to compute drift with a certain threshold and feature type (e.g., numerical or categorical). While there are other types of drift, such as concept drift and prediction drift, we only consider data drift monitoring in this initial version.

The *PrivacyMonitor* class represents a monitor that identifies privacy risks or leaks in the ML model by analyzing the labeled training dataset and production dataset with ground truth labels. This monitor also has attack models that can generate privacy attacks (e.g., black box membership inference attack) against the ML model to asses the privacy risks. This class contains one or more *PrivacyMetric* class instances with an enumeration attribute name for metric type (e.g., SHAPr membership privacy risk [24]) and a threshold.

Currently, the specification of two human-centric requirements (fairness and privacy) is supported in MoReML, however, the meta-model can be augmented to add support for additional human-centric requirements and associated metrics.

2.2.5 MonRequirement. This class represents the high-level requirements specifying which aspects of the system's behavior are being monitored for potential violations. These include human-centric requirements (e.g., fairness, privacy) and more technical requirements (e.g., performance and drift). We include this class in the meta-model for traceability between high-level requirements and the violations detected by runtime monitors. The class consists of an enumeration attribute type for the high-level requirement category and the specific requirement in natural language. Low-level technical monitoring requirements linked to these are specified as the metric classes.

2.2.6 Alert. This class describes the alert generated for ML engineers whenever a runtime monitor identifies a requirement violation. It contains a description of the alert and an email address.

2.3 MoReML Example

We designed MoReML, a textual DSL to enable the specification of an ML-based component, human-centric requirements, and runtime monitors. Listings 1, 2, 3 show snippets of a model created in MoReML. This model is based on an ML component that classifies whether the income of an adult exceeds $50K per year or not using features such as age, education, sex, marital status, and occupation.

Listing 1 shows the specification of a *MonitoringSystem* called demo (line 4) for an ML component with the id MLmodel-1 (line 5), a *Dataset* with ground truth labels (line 6-17), and a *MLcomponent* that uses a logistic regression model to classify adults' incomes (line 20-24). The dataset with model predictions and the performance

and privacy monitors are not shown in the example due to space limitations.

```
?nsuri: MLmonitoring_project
MonitoringSystem: {
    monsys_id: 1,
    name: demo,
    model: MLmodel-1,
    Dataset: {
        dataset_path: data/adult_labels.csv,
        label_name: income-per-year,
        column_names: [age, education-num, race, sex,
        capital-gain, capital-loss, hours-per-week,
        income-per-year, workclass, education,
        marital-status, occupation, relationship,
        native-country],
        categorical_features:[workclass, education,
        marital-status, occupation, sex, race,
        relationship]
    },
  .....
  }
MLcomponent: {
    model_id: MLmodel-1,
    algorithm: Logistic_Regression
}
```

Listing 1: MoReML model for the Monitoring System, ML Component and Dataset

Listing 2 depicts a *DriftMonitor* modeled for the 'education' feature (line 5). The monitor executes hourly (line 3) and calculates data drift using the probability stability index metric (line 7).

```
 DriftMonitor: {
        name: dmon1,
        interval: 3600, #1 hour in seconds
        active: True,
        features: [education],
        DriftMetric: {
            name: Probability_stability_index,
            threshold: 0.1,
            type: Categorical },
        MonRequirement: {
            type: Drift,
            requirement: Education feature must not
            drift more than 0.1) },
        Alert: {
            email: developedid@companyname.org,
            description: Data drift req. of MLmodel-1
            has been violated }
  }
```

Listing 2: MoReML model for Drift Monitor

Listing 3 shows a *FairnessMonitor* modeled for the protected attribute 'sex' (line 5) with the privileged group as 'male' (line 6). The monitor computes fairness in the predictions of the ML component using the statistical parity difference metric (line 8).

```
FairnessMonitor: {
        name: fmon1,
        interval: 86400, #24 hours in seconds
        active: True,
        protected_attributes: [sex],
        privileged_groups: [Male],
        FairnessMetric: {
            name: statistical_parity_difference,
            threshold: 0.1 },
        MonRequirement: {
            type: Fairness,
            requirement: Model must be fair w.r.t
            protected attribute sex (SPD should not
            be more than 0.1) },
        Alert: {
            email: developedid@companyname.org,
            description: Fairness req. of MLmodel-1 has
            been violated }
}
```

Listing 3: MoReML model for Fairness Monitor

2.4 Monitor Generation and Prototype Tool

ML experts model runtime monitors conforming to our meta-model shown in figure 3 without specifying any implementation details. The models are shown in Listing 1, 2, and 3 are defined in Flexmi [41], a reflective textual syntax for EMF models [67]. Other general textual syntaxes such as XML can also be used to define the models, however, we prefer the YAML syntax flavor of Flexmi [41]. To automatically generate runtime monitors we wrote a model-to-text transformation using the Epsilon Generation Language (EGL) [61]. The transformer takes models conforming to our meta-model as input and generates Python code for the runtime monitors. For infrastructure support, we leverage existing toolkits and libraries to implement metrics required for runtime monitoring. We use IBM AI fairness 360 [15] for fairness metrics, IBM AI adversarial robustness [58] for privacy metrics, Evidently AI [10] for drift metrics, and scikit-learn [45] for performance metrics. The generated code includes all necessary libraries, ensuring seamless integration without manual intervention. We implement our proof-of-concept prototype as an Eclipse-based [67] tool with our meta-model defined in Ecore and model-to-text transformations in Epsilon EGL [61].

3 EVALUATION

To demonstrate the utility of our approach, we carried out a preliminary evaluation using an ML component with a Logistic Regression model trained on the Adult dataset [1]. The model performs binary classification to determine if adult incomes exceed $50K per year or not using features such as age, education, sex, occupation, and weekly hours. We conduct an experiment to answer RQ1 and RQ2. For RQ1, we model ML monitoring aspects and related human-centric requirements in MoReML. Snippets of the model used in our evaluation are shown in Listings 1, 2, and 3. For RQ2, we intentionally introduced requirements violations to assess the performance of the generated monitors. We added dummy drift in the education feature for the drift monitor, leveraged a dataset with sex and race biases for the fairness monitor, omitted privacy risk mitigation for the privacy monitor, and set a high performance threshold for the performance monitor. Next, we divided the Adult dataset into training, testing, and production subsets, reserving one portion for assumed production data available in batch. The dataset is labeled, allowing us to use the labels as ground truth for metric calculation.

Our monitors were able to successfully identify the requirements violations. An example of our runtime ML monitor flagging requirements violations of drift and fairness is shown in Figure 4. The detected violation is presented with the monitor name, monitoring metric, value of the metric, and alert status.

```
_____________________________________
Violation Detected by Drift Monitor
_____________________________________
monitor name: dmon1
feature: education
metric: Probability_stability_index
value: 16.15
alert: sent

_____________________________________
Violation Detected by Fairness Monitor
_____________________________________
monitor name: fmon1
metric: statistical_partity_difference
value: 0.19
alert: sent
```

Figure 4: Violations Detected by Generated Monitors

4 LIMITATIONS

As our work is in its initial stages there are several limitations. Currently, our work is limited to i) a single ML component, ii) ML components for classification problems, iii) human-centric requirements of fairness and privacy, iv) a few monitoring metrics (all relying on the availability of ground truth labels), v) locally deployed ML components, vi) monitors receiving production data in batches, and vii) a preliminary evaluation. We have also not considered various ML frameworks (e.g. Tensorflow) in this version and would like to add support for them in the future.

5 RELATED WORK

Responsible ML: Responsible ML is a subfield of responsible artificial intelligence (AI) that refers to designing, developing, and deploying ML-based systems to benefit individuals, groups, and society while minimizing negative impacts [21]. Recently, many high-level responsible/ethical AI frameworks have been proposed by governments and large organizations including the European Union's Ethics Guidelines for Trustworthy AI [4], Australia's AI Ethics Principles [3], and more [11, 27, 39]. These frameworks emphasize that AI (including ML) systems must conform to human-centric requirements of fairness, privacy, explainability, accountability, safety, human values, and professional responsibility [27]. Despite the availability of several frameworks, none of them provide practical implementation guidance for responsible ML [63, 75]. Recently, some works have explored strategies to implement responsible ML in practice. For example, Sanderson et al. [63] interviewed AI practitioners to investigate the operationalization of responsible ML, they found governance, AI system design and development, competence and knowledge development, and stakeholder communication as the key practices required for realizing responsible AI. Zhu et al. [75] discuss methods to realize responsible AI by achieving trustworthiness through product and process assurance mechanisms. While many studies on responsible ML mention the importance and need for runtime monitoring [50, 75], none provide detailed implementable solutions.

Runtime Monitoring of ML Systems: Numerous runtime monitoring solutions for ML-based systems are available that monitor various requirements violations. Kourouklidis et al. [43] propose a low-code approach for performance monitoring of ML using various drift detection techniques. A domain-specific language is described in [44] for data scientists to specify ML model monitoring workflows and a runtime component for software engineers that implements the monitoring behavior. An approach and toolset for realistic data drift detection in deployed ML systems is proposed in [48], the approach focuses on drift behavior analysis and informed monitoring for responsible AI. An ML model monitoring solution is described in [73], it detects anomalies in model inputs and predictions by leveraging deep learning. Other examples of studies on drift and anomaly monitoring include [20, 36, 59], and [70]. Some prominent commercial and open-source tools for runtime monitoring of ML-based systems include Amazon SageMaker [9], IBM Watson OpenScale [6], Microsoft Azure [8], Google Vertex AI [7], Arize AI [2], and Evidently AI [10]. While most studies and tools focus on monitoring technical requirements violations in ML-based systems, few studies explore monitoring for human-centric requirements violations. These include fairness and bias monitoring [13, 29, 37, 38], privacy and adversarial robustness monitoring [40, 54], safety monitoring [14, 34, 68, 71], and trust monitoring [18, 25, 46, 62]. Few approaches for monitoring human-centric requirements violations exist. The ones that do have several limitations: 1) Studies focus only on a single human-centric requirement [13, 46, 59]. 2) Technical expertise in statistics and software engineering are required to set up a monitoring system [43, 59]. 3) Limited support for monitoring evolving human-centric requirements. As for commercial solutions, they are platform-dependent, require payment for most services, and do not consider human-centric requirements other than fairness and explainability [2, 7, 9].

6 CONCLUSION AND FUTURE WORK

We have described a novel MDE approach for runtime monitoring of ML components to adhere to human-centric requirements, thus ensuring responsible behavior. Our approach successfully abstracts and automates the development and maintenance of runtime monitors for human-centric requirements violations. We proposed a DSL, MoReML, to represent human-centric requirements, ML components, and runtime monitors. We also provided an example of a MoReML model, demonstrating how to specify these in an Eclipse-based tool. Initial evaluation of our toolset and approach shows that it can model ML components for classification problems, human-centric requirements of fairness and privacy, and generate runtime monitors using existing toolkits.

A number of future work directions exist. These include adding support for a wider range of ML components and human-centric requirements, monitoring multiple ML components, generating a wider set of ML monitors, establishing connections between technical aspects and human-centric requirements (e.g., the impact of drift on privacy), and incorporating metrics that do not depend on ground truth labels. We also plan to extensively evaluate the usability and usefulness of MoReML and our approach in wider ML-heavy applications.

ACKNOWLEDGMENTS

Naveed is supported by a Faculty of IT Post-graduate scholarship. Grundy and Haggag are supported by ARC Laureate Fellowship FL190100035. This work is also partly supported by ARC Discovery Project DP200100020.

REFERENCES

[1] [n. d.]. *Adult dataset.* https://archive.ics.uci.edu/ml/datasets/Adult
[2] [n. d.]. *The AI Observability & LLM Evaluation Platform.* https://arize.com/
[3] [n. d.]. *Australia's AI Ethics Principles.* https://www.industry.gov.au/publications/australias-artificial-intelligence-ethics-framework/australias-ai-ethics-principles
[4] [n. d.]. *Ethics guidelines for trustworthy AI.* https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai
[5] [n. d.]. *Future of ai technologies report.* https://www.gartner.com/smarterwithgartner/gartner-predictsthe-future-of-ai-technologies
[6] [n. d.]. *IBM Watson Openscale.* https://www.ibm.com/docs/en/cloud-paks/cp-data/3.5.0?topic=services-watson-openscale
[7] [n. d.]. *Introduction to Vertex AI Model Monitoring.* https://cloud.google.com/vertex-ai/docs/model-monitoring/overview
[8] [n. d.]. *Model monitoring with Azure Machine Learning.* https://learn.microsoft.com/en-us/azure/machine-learning/concept-model-monitoring?view=azureml-api-2
[9] [n. d.]. *Monitor data and model quality.* https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html
[10] [n. d.]. *The open-source ML observability platform.* https://www.evidentlyai.com/
[11] [n. d.]. *Principles for the Ethical Use of AI in the UN System.* https://unsceb.org/principles-ethical-use-artificial-intelligence-united-nations-system
[12] Khlood Ahmad, Mohamed Abdelrazek, Chetan Arora, Muneera Bano, and John Grundy. 2023. Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology* (2023), 107176.
[13] Aws Albarghouthi and Samuel Vinitsky. 2019. Fairness-aware programming. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 211–219.
[14] Koorosh Aslansefat, Ioannis Sorokos, Declan Whiting, Ramin Tavakoli Kolagari, and Yiannis Papadopoulos. 2020. SafeML: safety monitoring of machine learning classifiers through statistical difference measures. In *International Symposium on Model-Based Safety and Assessment*. Springer, 197–211.
[15] Rachel KE Bellamy, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilović, et al. 2019. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development* 63, 4/5 (2019), 4–1.
[16] Thomas Brand and Holger Giese. 2018. Towards software architecture runtime models for continuous adaptive monitoring.. In *MoDELS (Workshops)*. 72–77.
[17] Petter Bae Brandtzaeg and Asbjørn Følstad. 2018. Chatbots: changing user needs and motivations. *interactions* 25, 5 (2018), 38–43.
[18] Taejoon Byun and Sanjai Rayadurgam. 2020. Manifold for machine learning assurance. In *ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 97–100.
[19] Australia's National Artificial Intelligence Centre. [n. d.]. *Responsible AI.* https://www.csiro.au/en/work-with-us/industries/technology/national-ai-centre,2023.Accessed:November2023
[20] Oliver Cobb and Arnaud Van Looveren. 2022. Context-aware drift detection. In *International Conference on Machine Learning*. PMLR, 4087–4111.
[21] CSIRO. 2023. *Responsible AI Pattern Catalogue.* https://www.csiro.au/en/research/technology-space/ai/Responsible-AI/RAI-Pattern-Catalogue
[22] Alex Cummaudo, Scott Barnett, Rajesh Vasa, and John Grundy. 2020. Threshy: Supporting safe usage of intelligent web services. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1645–1649.
[23] Jeffrey Dastin. 2022. Amazon scraps secret AI recruiting tool that showed bias against women. In *Ethics of data and analytics*. Auerbach Publications, 296–299.
[24] Vasisht Duddu, Sebastian Szyller, and N Asokan. 2021. SHAPr: An efficient and versatile membership privacy risk metric for machine learning. *arXiv preprint arXiv:2112.02230* (2021).
[25] Lisa Ehrlinger, Verena Haunschmid, Davide Palazzini, and Christian Lettner. 2019. A DaQL to monitor data quality in machine learning applications. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30*. Springer, 227–237.
[26] Hans-Erik Eriksson, Magnus Penker, Brian Lyons, and David Fado. 2003. *UML 2 toolkit*. John Wiley & Sons.
[27] Jessica Fjeld, Nele Achten, Hannah Hilligoss, A Nagy, and Madhulika Srikumar. 2020. Principled artificial intelligence. *Berkman Klein Center, February* 14 (2020).
[28] Robert France and Bernhard Rumpe. 2005. Domain specific modeling. *Software & Systems Modeling* 4, 1 (2005), 1–3.
[29] Avijit Ghosh, Aalok Shanbhag, and Christo Wilson. 2022. Faircanary: Rapid continuous explainable fairness. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*. 307–316.
[30] Tony Ginart, Martin Jinye Zhang, and James Zou. 2022. Mldemon: Deployment monitoring for machine learning systems. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3962–3997.
[31] Polyxeni Gkontra, Gianluca Quaglio, Anna Tselioudis Garmendia, and Karim Lekadir. 2023. Challenges of Machine Learning and AI (What Is Next?), Responsible and Ethical AI. In *Clinical Applications of Artificial Intelligence in Real-World Data*. Springer, 263–285.
[32] Johannes Grohmann, Patrick K Nicholson, Jesus Omana Iglesias, Samuel Kounev, and Diego Lugones. 2019. Monitorless: Predicting performance degradation in cloud applications with machine learning. In *Proceedings of the 20th international middleware conference*. 149–162.
[33] John Grundy, Hourieh Khalajzadeh, Jennifer McIntosh, Tanjila Kanij, and Ingo Mueller. 2020. Humanise: Approaches to achieve more human-centric software engineering. In *International Conference on Evaluation of Novel Approaches to Software Engineering*. Springer, 444–468.
[34] Joris Guerin, Raul Sena Ferreira, Kevin Delmas, and Jérémie Guiochet. 2022. Unifying evaluation of machine learning safety monitors. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 414–422.
[35] Thilo Hagendorff and David Danks. 2023. Ethical and methodological challenges in building morally informed AI systems. *AI and Ethics* 3, 2 (2023), 553–566.
[36] Ben Halstead, Yun Sing Koh, Patricia Riddle, Russel Pears, Mykola Pechenizkiy, Albert Bifet, Gustavo Olivares, and Guy Coulson. 2022. Analyzing and repairing concept drift adaptation in data stream classification. *Machine Learning* 111, 10 (2022), 3489–3523.
[37] Michaela Hardt, Xiaoguang Chen, Xiaoyi Cheng, Michele Donini, Jason Gelman, Satish Gollaprolu, John He, Pedro Larroy, Xinyu Liu, Nick McCarthy, et al. 2021. Amazon sagemaker clarify: Machine learning bias detection and explainability in the cloud. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2974–2983.
[38] Thomas Henzinger, Mahyar Karimi, Konstantin Kueffner, and Kaushik Mallik. 2023. Runtime Monitoring of Dynamic Fairness Properties. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*. 604–614.
[39] Anna Jobin, Marcello Ienca, and Effy Vayena. 2019. The global landscape of AI ethics guidelines. *Nature machine intelligence* 1, 9 (2019), 389–399.
[40] Myeongseob Ko, Xinyu Yang, Zhengjie Ji, Hoang Anh Just, Peng Gao, Anoop Kumar, and Ruoxi Jia. 2023. PrivMon: A Stream-Based System for Real-Time Privacy Attack Detection for Machine Learning Models. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 264–281.
[41] Dimitris Kolovos and Alfonso de la Vega. 2023. Flexmi: a generic and modular textual syntax for domain-specific modelling. *Software and Systems Modeling* 22, 4 (2023), 1197–1215.
[42] Dimitrios S Kolovos, Richard F Paige, and Fiona AC Polack. 2009. On the evolution of OCL for capturing structural constraints in modelling languages. *Rigorous Methods for Software Construction and Analysis: Essays Dedicated to Egon Börger on the Occasion of His 60th Birthday* (2009), 204–218.
[43] Panagiotis Kourouklidis, Dimitris Kolovos, Joost Noppen, and Nicholas Matragkas. 2021. A model-driven engineering approach for monitoring machine learning models. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 160–164.
[44] Panagiotis Kourouklidis, Dimitris Kolovos, Joost Noppen, and Nicholas Matragkas. 2023. A domain-specific language for monitoring ML model performance. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 266–275.
[45] Oliver Kramer and Oliver Kramer. 2016. Scikit-learn. *Machine learning for evolution strategies* (2016), 45–53.
[46] Michael Austin Langford, Kenneth H Chan, Jonathon Emil Fleck, Philip K McKinley, and Betty HC Cheng. 2021. Modalas: Model-driven assurance for learning-enabled autonomous systems. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 182–193.
[47] Dorian Leroy, Pierre Jeanjean, Erwan Bousse, Manuel Wimmer, and Benoit Combemale. 2020. Runtime monitoring for executable DSLs. *The Journal of Object Technology* 19, 2 (2020), 1–23.
[48] Grace A Lewis, Sebastián Echeverría, Lena Pons, and Jeffrey Chrabaszcz. 2022. Augur: A step towards realistic drift detection in production ml systems. In *Proceedings of the 1st Workshop on Software Engineering for Responsible AI*. 37–44.
[49] Lydia T Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. 2018. Delayed impact of fair machine learning. In *International Conference on Machine Learning*. PMLR, 3150–3158.
[50] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, and Zhenchang Xing. 2022. Towards a roadmap on software engineering for responsible AI. In *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*. 101–112.
[51] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, Didar Zowghi, and Aurelie Jacquet. 2023. Responsible ai pattern catalogue: A collection of best practices for ai governance and engineering. *Comput. Surveys* (2023).
[52] Lucy Ellen Lwakatare, Aiswarya Raj, Ivica Crnkovic, Jan Bosch, and Helena Holmström Olsson. 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology* 127 (2020), 106368.

[53] Andrew McNamara, Justin Smith, and Emerson Murphy-Hill. 2018. Does ACM's code of ethics change ethical decision making in software development?. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 729–733.

[54] Seung Ho Na, Kwanwoo Kim, and Seungwon Shin. 2023. Witnessing Erosion of Membership Inference Defenses: Understanding Effects of Data Drift in Membership Privacy. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 250–263.

[55] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration challenges in building ml-enabled systems: Communication, documentation, engineering, and process. In *Proceedings of the 44th international conference on software engineering*. 413–425.

[56] Hira Naveed. 2023. Runtime Monitoring of Human-Centric Requirements in Machine Learning Components: A Model-Driven Engineering Approach. In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 146–152.

[57] Hira Naveed, Chetan Arora, Hourieh Khalajzadeh, John Grundy, and Omar Haggag. 2024. Model driven engineering for machine learning components: A systematic literature review. *Information and Software Technology* (2024), 107423.

[58] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. 2018. Adversarial Robustness Toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069* (2018).

[59] David Nigenda, Zohar Karnin, Muhammad Bilal Zafar, Raghu Ramesha, Alan Tan, Michele Donini, and Krishnaram Kenthapadi. 2022. Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3671–3681.

[60] Theresa Roland, Carl Böck, Thomas Tschoellitsch, Alexander Maletzky, Sepp Hochreiter, Jens Meier, and Günter Klambauer. 2022. Domain shifts in machine learning based Covid-19 diagnosis from blood tests. *Journal of Medical Systems* 46, 5 (2022), 23.

[61] Louis M Rose, Richard F Paige, Dimitrios S Kolovos, and Fiona AC Polack. 2008. The epsilon generation language. In *Model Driven Architecture–Foundations and Applications: 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, 2008. Proceedings 4*. Springer, 1–16.

[62] Anirban Roy, Adam Cobb, Nathaniel Bastian, Brian Jalaian, and Susmit Jha. 2022. Runtime monitoring of deep neural networks using top-down context models inspired by predictive processing and dual process theory. In *AAAI Spring Symposium 2022*.

[63] Conrad Sanderson, Qinghua Lu, David Douglas, Xiwei Xu, Liming Zhu, and Jon Whittle. 2022. Towards implementing responsible AI. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 5076–5081.

[64] Tim Schröder and Michael Schulz. 2022. Monitoring machine learning models: a categorization of challenges and methods. *Data Science and Management* 5, 3 (2022), 105–116.

[65] Zhihui Shao and Jianyi Yang. 2020. Increasing the Trustworthiness of Deep Neural Networks via Accuracy Monitoring. In *Workshop on Artificial Intelligence Safety 2020 (co-located with IJCAI-PRICAI 2020)*.

[66] Pravik Solanki, John Grundy, and Waqar Hussain. 2023. Operationalising ethics in artificial intelligence for healthcare: A framework for AI developers. *AI and Ethics* 3, 1 (2023), 223–240.

[67] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. 2008. *EMF: eclipse modeling framework*. Pearson Education.

[68] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. 2020. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*. 359–371.

[69] Eliza Strickland. 2019. IBM Watson, heal thyself: How IBM overpromised and underdelivered on AI health care. *IEEE Spectrum* 56, 4 (2019), 24–31.

[70] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. 2022. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*. PMLR, 20827–20840.

[71] Hazem Torfah and Sanjit A Seshia. [n. d.]. Runtime Monitors for Operational Design Domains of Black-Box ML-Models. In *NeurIPS ML Safety Workshop*.

[72] Michael Vierhauser, Antonio Garmendia, Marco Stadler, Manuel Wimmer, and Jane Cleland-Huang. 2023. GRuM—A flexible model-driven runtime monitoring framework and its application to automated aerial and ground vehicles. *Journal of Systems and Software* 203 (2023), 111733.

[73] Zhentao Xu, Ruoying Wang, Girish Balaji, Manas Bundele, Xiaofei Liu, Leo Liu, and Tie Wang. 2023. Alertiger: Deep learning for ai model health monitoring at linkedin. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5350–5359.

[74] Alfa Yohannis and Dimitris Kolovos. 2022. Towards model-based bias mitigation in machine learning. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. 143–153.

[75] Liming Zhu, Xiwei Xu, Qinghua Lu, Guido Governatori, and Jon Whittle. 2022. AI and ethics—Operationalizing responsible AI. *Humanity driven AI: Productivity, well-being, sustainability and partnership* (2022), 15–33.

Received 28 March 2024