TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 66, Issue III, August, 2023

FROM DATA TO DECISIONS: THE IMPORTANCE OF MONITORING ML SYSTEMS IN INDUSTRIAL SETTINGS

Adrian-Ioan ARGESANU, Gheorghe-Daniel ANDREESCU

Abstract: *Machine learning (ML) systems have been extensively used in various industrial applications, including process optimization, predictive maintenance, quality control, and decision support. The success of ML systems in these domains depends on their ability to make accurate predictions, decisions, and recommendations. With all ML models being inherently perishable, precautionary measures need to be put in place to ensure that degrading performance is detected before it yields negative outcomes. In this paper, we explore the importance of monitoring ML systems, and the various areas that need to be surveilled. We present a novel implementation, leveraging a bespoke combination of components to achieve complete visibility over the state of the ML solution at any given moment, discussing employed methods and how these provide insights for Process & Pipeline Services' use-case of detecting mechanically induced stress cracking in pipelines. This paper adds to the limited literature on ML system monitoring.*
Key words: *AI, machine learning, system monitoring, prediction monitoring, data monitoring.*

1. INTRODUCTION

Machine learning has found widespread adoption in various fields of the industrial sector, such as manufacturing, logistics, and energy. ML systems use large datasets to construct models that make predictions or decisions in either real-time or batch-mode. Machine learning has been shown to outperform traditional approaches in many cases, leading to improved efficiency, accuracy, and cost savings.

Situated at the core of ML systems, the ML models are known to be inherently perishable, meaning that their performance can decline over time. Precautionary measures are therefore required, to ensure that degrading performance is detected before it yields negative outcomes.

When compared to traditional software systems, ML systems exhibit all the challenges of traditional code, plus an array of machine learning-specific considerations [6]. Unlike in traditional software systems, an ML system's behavior is governed not just by rules specified in the code, but also by model behavior learned from data. This presents a major challenge as the real-world conditions where ML systems are deployed may change over time [5], causing shifts in the data distribution, the introduction of new types of events or inputs, and the emergence of new correlations or dependencies. These changes may affect the performance of the ML system and make it less accurate, relevant, and reliable.

One way of tackling this is to surveil the performance of ML systems and detect any deviations from the expected performance. This can be done by regularly collecting performance metrics, such as accuracy, recall, precision, F1-score, and AUC, and comparing them to the performance metrics obtained during the training phase.

In addition, it is important to monitor the inputs, features, and outputs of the ML system and detect any anomalies, outliers, or discrepancies. This can be done by using statistical or graphical methods, such as histograms, box plots, or scatter plots, or by using machine learning methods, such as anomaly detection or clustering.

Tracking system availability is crucial in ensuring ML offerings remain accessible to their

Received: 28.06.23; **Similarities**: 04.07.23: **Reviewed**: 14.07./24.07.23: **Accepted**:20.08.23.

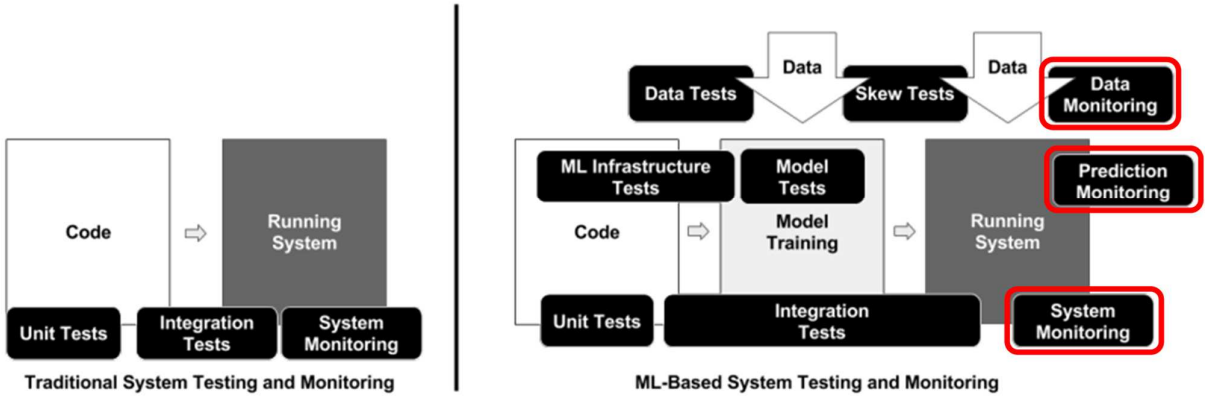

Fig. 1. Monitoring of traditional code (left) versus ML systems (right) [1].

consumers, and dependent business processes can continue without interruption.

As ML system can change over time, regular updates and maintenance of the monitoring setup are also necessary to ensure that the system stays relevant. Regular monitoring also provides an opportunity for identifying areas for improvement, enabling ongoing optimization of the system.

Despite the complexity and importance in operational excellence, there is limited literature discussing ML monitoring aspects and solutions [10]. [4] highlights the importance of ML system monitoring in the context of 3 failure scenarios: hard – such as data source failures, soft – such as feature drift in the data pipeline, and drift errors – referring to the natural data drift. Shankar et al. [7] note that many ML-related interview studies exist, but these focus primarily on engineering, not operational, challenges. Surveys such as [9] recognize a trend towards interactive data visualization tools supporting decision makers. [10] concludes that ML monitoring needs to be considered an augmentation to the existing well maintained software service monitoring practices.

This paper aims to help close this gap in literature. The presented concepts and methods, although applied to a production-deployed industrial ML application, are applicable to any ML system of any domain throughout its entire lifecycle.

In the following section we discuss relevant areas of monitoring for ML systems, providing recommendations for each of them. Section 3 introduces practical implementations for all presented aspects, showcasing employed methods and resulting insights on Process & Pipeline Services' use-case of detecting mechanically induced stress cracking in pipelines. Section 4 concludes by summarizing ML system monitoring importance, outlining possible directions for future work.

2. MONITORING AREAS AND BEST PRACTICES

Monitoring can be defined as continuous observation of performance and behavior, with the aim of detecting any issues or deviations from the expected outcome.

For machine learning systems in specific, monitoring refers to the way performance is tracked and understood from both a data science and operational perspective. Figure 1 compares traditional software systems to ML ones, highlighting the additional challenges pertaining to the post-production stage of the ML lifecycle. We include "System Monitoring" due to the increased overall complexity of the resulting system.

Data monitoring plays a vital role as it ensures that the prediction candidates are compatible with the model, or in other words, the model is certified to accurately infer results on the provided input data. In case of incompatibility, the inference outcome is undefined and should not be trusted. This is of extreme importance for safety-critical applications.

Data monitoring involves observing various aspects of data such as missing values, outliers,

or changes in data distributions. By detecting these issues, data monitoring helps to prevent the ML system from producing incorrect results or failing entirely.

Anomalies detected as part of data monitoring can typically be translated into pre-predict conditions, which act as filters, disqualifying data before the inference step.

Post inference, the predictions, confidence scores, as well as any exceptional circumstance that prevented processing of specific data items must be kept track of. Referred to as **prediction monitoring**, this helps to detect any anomalies, such as bias or inconsistency, and guarantees that the ML system is continuing to make accurate predictions when compared to the performance achieved during training. In addition, ML features can be captured for detailed analysis whenever model introspection is required.

Prediction monitoring is best aligned with the performance goals specific to the requirements and expectations of the application. These goals should be defined before deployment and used as the basis for selecting appropriate performance metrics, such as accuracy, recall, precision, F1-score, AUC, or any combinations of the these. For industrial or safety-critical applications, conservative-skewed metrics are preferred, ensuring safety-outliers are eliminated or at least minimized – e.g., higher FP rates might be acceptable if the FN rate is kept close to 0.

For applications where ML systems are integrated in complex data pipelines, prediction monitoring is of exceptional importance, as incorrect predictions can lead to operational problems, financial losses, and potential safety hazards.

Prediction monitoring is imperative for the definition of post-prediction safeguards – when circumstances render the inference process "best-endeavor" or predictions need to be invalidated, as well as for strategic ML lifecycle decisions such as model finetuning or retraining – e.g., when performance degradation is detected.

System monitoring refers to the monitoring of the ML system as a whole. This encompasses the monitoring of hardware indicators such as CPU and memory utilization, software performance indicators such as processing time, as well as identifying potential problems with the system itself. In production settings, system monitoring is essential for predictive maintenance and recovery from hardware and software failures, thus downtime reduction and minimization of potential operational impact.

Data, **prediction**, and **system monitoring** are essential for ensuring smooth and effective operation of ML applications. As the applications evolve over time, the monitoring systems and processes might lose effectiveness. To counteract that, it is necessary to design these to continuously adapt to changes in the system and data. Moreover, regular review and evaluation of the monitoring processes is necessary to guarantee their continued robustness and relevance.

The involvement of human experts in the monitoring process is also a key factor. Their expertise and judgment can provide valuable assistance in situations where the process does not provide a clear recommendation. The combination of human and automated monitoring can guarantee that the ML system is running at its best, providing accurate predictions.

With monitoring systems capturing significant amounts of data, interpretation and visualization of the surveillance results becomes a challenge. The data can be complex, multi-dimensional, and difficult to understand, especially for non-experts or stakeholders who are not familiar with the underlying concepts and methods. To address this challenge, it is necessary to develop user-friendly, interactive, and accessible visualization tools and dashboards that allow various user groups to explore and interpret the results in a meaningful and intuitive way.

In summary, maintaining the effectiveness of machine learning models requires ongoing effort and attention. By combining automated monitoring, human expertise, regular evaluations and updates, and a clear understanding of the business requirements and model limitations, organizations can ensure that their models remain effective, safe, and continue to deliver positive outcomes.

3. CASE STUDY

For a practical implementation of the presented concepts and considerations, we focus on Process & Pipeline Services' pipeline inspection use-case. With pipelines considered the primary means to transport oil or gas safely and efficiently at high pressures across long distances, the safety and availability of the pipeline system is a primary concern. From the integrity perspective pipelines can be treated as pressure vessels. These are particularly susceptible to longitudinal cracking due to circumferential stress. Process & Pipeline Services' UltraScan™ CD Plus [8] crack detection tool has been developed to detect and identify any anomaly of the pipeline wall before it has a detrimental effect on the integrity of a given line or system.

We employ our Platform for End-to-End Lifecycle Management of Batch-Prediction Machine Learning Models, introduced in [2], to support the safety-critical analysis of the 4-dimensional signal data recorded by the UltraScan™ CD Plus inspection tool via a complex ML model.

In our platform, ML models are hosted in the form of Docker containers. Environments –the collection of all deployed models, their auxiliary services such as data ingress/egress connectors, and management services, are orchestrated via Docker Swarm. Each environment features a middleware storage service, which persists all prediction requests, errors, and outcomes.

In our ML monitoring solution, we employ a multi-tiered setup. The novelty of the approach lies in how the individual components are selected, layered, integrated, and presented to various levels of stakeholders. We screen and capture:

- Availability of hardware and its usage (HW uptime and utilization) as part of system monitoring.
- Availability of individual services and their status (service uptime and health) as part of system monitoring.
- Performance of machine learning models (input and safeguard monitoring, prediction outcomes, feedback loops) as a combination of data and prediction monitoring.
- Event logs from all models and services (auditability and debugging) as a combination of all three monitoring areas highlighted in Figure 1.

Fig. 2. UltraScan™ CD Plus [8] (top), with mechanical stress induced cracking examples – fatigue, stress-corrosion-cracking (SCC) and shrinkage (bottom).

For availability of hardware and services we rely on metrics. Metrics represent the raw measurements of resource usage or behavior, that can be observed and collected throughout the systems. These might be low-level usage summaries provided by the operating system, or higher-level types of data tied to the specific functionality or work of a component, like requests served per second or outputs from a particular function. For metric scraping, aggregation, visualization and interpretation, we employ a Prometheus + Grafana setup (Figure 3):

- We run a Prometheus Node Exporter instance on each node to expose a wide variety of hardware- and kernel-related metrics, including, but not limited to overall usage of CPU, RAM, and disk space.
- Each node also hosts a Google cAdvisor (Container Advisor) daemon that collects, aggregates, processes, and exports information about running containers. For each container, cAdvisor tracks resource isolation parameters, histograms of complete historical resource usage and network statistics.
- Furthermore, we deploy custom metric scrapers in the form of Docker containers on each node to monitor service health data via the health APIs of the deployed services.
- The metrics scraped by all of these containers are aggregated and stored as timeseries data in a central Prometheus instance.

- To visualize, monitor and analyze all data collected by Prometheus, we employ the observability tool Grafana.

Stakeholders, as well as members of the DevOps team use Grafana as an entry point when assessing the availability and health of the overall application or system. Grafana supports the definition of visual alerts, to allow users to quickly home in on problematic areas. For our platform, we have configured alerts for:

- Failing services. We consider a service as failed whenever it returns a negative or no result during a health check. Failed services need immediate attention to ensure application recovery.
- Predictive maintenance. In this category we alert e.g., for low available disk space. This helps plan maintenance windows upfront, minimizing production outage.
- System optimization potential. Tracking hardware metrics over time allows the detection of e.g., disproportionally high CPU usage during certain operations or in certain parts of the distributed system. Optimization efforts can subsequently be planned on mid- and long-term time horizons to improve overall system performance.

Figure 4 displays an excerpt of the Grafana dashboard and tracks the availability of individual hardware nodes – or instances, and services. The "Model Service Health" widget displayed in the bottom half of the figure shows the data collected in Prometheus from the custom health-metric scrapers of the ML models. The result of the health-check is displayed in binary format, where 1 means the service is healthy. The graph shows intermittent health check failures, and the associated alerts,

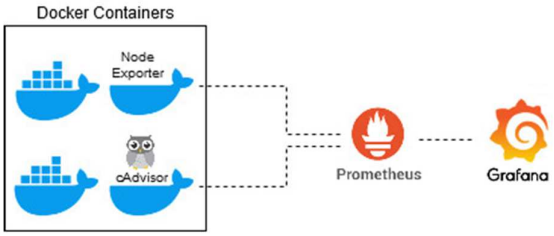

Fig. 3. Hardware and service monitoring with Prometheus and Grafana.

Fig. 4. Grafana dashboard – instance and service health.

around 18:00, all of which were immediately recovered from.

The "Running Instances" graph tracks, at the most basic of levels, the availability of the hardware nodes hosting various services. Results are again depicted in binary fashion, where 1 means the machine is online.

For prediction and data monitoring, we leverage the data persisted in a middleware storage. Currently in the form of a relational database, the middleware storage is interacted with before and after each prediction request to persist all prediction candidates and update them with relevant ML features as computed by the model's data pipeline, as well as inference results and exceptional situations.

For visual assessment of the data of the middleware store, we offer a Redash view. Dashboards can be configured, saved, and shared with various user groups in Redash. For each deployed model, we recommend a dedicated dashboard containing the following minimum set of tiles:

- Automation percentage(s) (number of predictions with high confidence scores versus total candidates) – one tile per model output.
- Prediction score distribution(s) – one tile per model output.
- Pre-predict filter statistics.
- Post-predict safeguard statistics.

Redash can be linked up with all data sources the deployed platform has access to, which

enables the creation of feedback dashboards. For applications where human experts engage in QA checks of inference results to potentially overrule these, Redash can be configured to display overruled predictions side-by-side with the middleware data, enabling expert personnel to detect shortcomings of the ML model. This feature is extensively leveraged in Process & Pipeline Services' use-case to track model performance and ensure it remains safe to use.

Figure 5 depicts the dashboard of Process & Pipeline Services' ML system for detection of mechanically induced stress cracking in pipelines. The model is tasked with identifying two patterns in the data, referred to as "event 1" and "event 2". The model is configured to run in batch-mode; the depicted results are for one dataset. The top half of the figure depicts the automation percentage per event (left hand side) as well as the inference score distribution (right hand side). Qualified personnel can evaluate this information for abnormal automation rates – too high or too low, as well as population segregation – in case of classification models.

The bottom half of Figure 5 presents a summary of the pre-predict filters, followed by a summary of the post-predict safeguards. One example of a pre-predict filter is a compatibility assertion between the input vector of the prediction candidate and the model contract – e.g., a value-range constraint. A second example

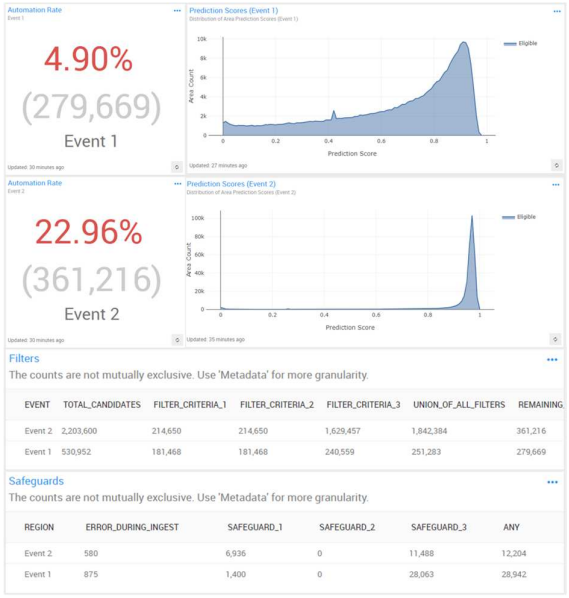

Fig. 5. Redash dashboard.

checks whether the input vector was successfully processed by all stages of the data pipeline. Any failing pre-predict check is immediately noted in the protocol associated with the prediction candidate and the candidate is disqualified from inference, mitigating safety-outliers.

The post-predict safeguards are executed after the model has predicted on the candidate. The main goal is to notify the requester if any of the predictions need to be treated as "best-endeavor". One example safeguard for the presented system evaluates whenever the 4D input vector of a prediction candidate was cropped to match the input requirements of the model. In such scenarios, the outcome of the prediction might still be relevant for augmentation or QA scenarios, but is disqualified for safety-critical automation applications. Any post-predict failure is noted in the protocol of the prediction candidate. The client is responsible for acting on safeguard hits. The depicted model applies the same pre-prediction filters and post-prediction safeguards for both events.

Whenever the pre- or post-predict statistics highlight anomalous behavior, a detailed analysis of the chain of events might be required. For aggregated interpretation of events from different services, potentially deployed across multiple machines, we offer a Kibana interface. As part of the ELK stack (Elasticsearch + Logstash + Kibana), Kibana (the flexible visualization tool), sits on top of Elasticsearch (the open source, distributed, RESTful, JSON-based search engine), which gets fed by Logstash (the ingest pipeline) from several Beats instances (lightweight, single-purpose data shippers). The Beats instances tail the logs of all running services, thus allowing temporal correlation in Kibana. Kibana also allows users to search through historic logs for additional occurrences of events of interest. Figure 6 presents the ELK stack, showcasing how Docker log messages are aggregated to a log-level frequency bar chart.

In summary, our monitoring solution offers three dashboards, each catering to a specific level of detail and thus audience. With Grafana as the high-level system view, stakeholders and maintenance personnel have a one-stop

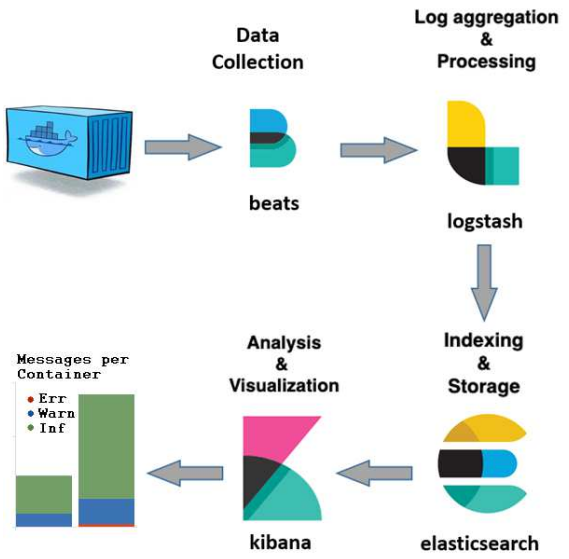

Fig. 6. ELK stack.

dashboard for the health of the application. Leveraging all historic prediction data, Redash can be used to display high-level as well as detailed insights to stakeholders and expert users alike, enabling them to track the behavior of each ML model versus its performance targets. Finally, the Kibana dashboard allows temporal correlation of all events recorded throughout the system, greatly improving failure analysis and debugging activities for development personnel, whilst eliminating the need to manually aggregate information from different sources.

All components of the monitoring setup have been chosen for their ability to adapt to the changing needs of ML systems. For collection of additional data, further metrics can be programmed in custom scraping containers, ML model contracts can be extended to forward more insights, extended logging can be implemented for improved debugging. For persistence of this data, Prometheus can be configured to scrape the new metrics, and the schema of the middleware storage can be extended to capture the additional prediction artefacts; the ELK stack does not require any updates, as it implicitly captures all available information. Visualization is subsequently enabled by Grafana, Redash and Kibana; their dashboards can be freely configured to present the collected data in meaningful ways.

Combining the robustness of automated monitoring with the sophistication of applying of human expertise through regular evaluation of the complex collected insights, our proposed solution enables Process & Pipeline Services' to reliably operate its safety-critical ML system for mechanically induced stress cracking of pipelines, with close to 0 downtime across multiple years of operation.

4. CONCLUSIONS AND FUTURE WORK

Monitoring ML systems is a complex task that requires careful planning, design, and implementation. The challenges of monitoring ML systems include the selection of appropriate metrics, regular collection of performance data, the interpretation and visualization of the results, and the maintenance and updating of the monitoring frameworks. These are all accentuated for industrial and safety-critical applications, where inaccurate predictions can lead to operational problems, financial losses, and safety hazards.

Our bespoke multi-tiered approach to monitoring addresses the evolving needs of data, prediction as well as system monitoring, providing insights to a variety of user groups – from technical experts to stakeholders who are not familiar with ML concepts and methods. By implementing strong monitoring processes, incorporating human expertise, and addressing potential implications, we ensure the optimal operation of ML systems and the generation of reliable predictions under real-world scenarios, ads demonstrated for Process & Pipeline Services' use-case.

As much as monitoring can be automated, interpretation of complex graphs and statistics remains largely a manual task carried out by human experts. To reduce the workload and associated costs, future work will focus on the development of automated surveillance and maintenance tools, such as self-diagnostic systems and auto-tune methods.

Furthermore, with the increasing appetite for explainability of ML decisions, future research will focus on investigating new methods and means to increase the transparency and accountability of ML systems, and their integration into monitoring solutions.

This paper adds to the limited literature on ML system monitoring.

5. REFERENCES

[1] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley, *The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction*, Proceedings of IEEE Big Data, 2017.

[2] A. -I. Argesanu, G. -D. Andreescu, *A Platform to Manage the End-to-End Lifecycle of Batch-Prediction Machine Learning Models*, 2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), 2021, pp. 329-334.

[3] C. Sridharan, *Distributed Systems Observability*, O'Reilly Media, Inc., 2018, ISBN 9781492033424.

[4] S. Shankar, A. Parameswara, *Towards Observability for Machine Learning Pipelines*, 2021, https://arxiv.org/abs/2108.13557.

[5] Y. Wu, E.D. Yinjun, S.B. Davidson. *DeltaGrad: Rapid retraining of machine learning models*, International Conference on Machine Learning, 2020.

[6] E.D. Nascimento, I., Ahmed, E. Oliveira, M.P. Palheta, I. Steinmacher, T.U. Conte, *Understanding Development Process of Machine Learning Systems: Challenges and Solutions*, 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 1-6, 2019.

[7] S. Shankar, R. Garcia, J.M. Hellerstein, A.G. Parameswaran. *Operationalizing machine learning: An interview study*, arXiv preprint arXiv:2209.09125 (2022).

[8] *Crack capabilities overview*, Process & Pipeline Services, accessed June 2023, https://www.bakerhughes.com/sites/bakerhughes/files/2020-07/19004_BH_PPS_ILI_US_BRO_1912%20%28CRACK%20CAPAB%29.pdf

[9] G. Nguyen, S. Dlugolinsky, M. Bobák, et al. *Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey*, Artif Intell Rev 52, 77–124 (2019), https://doi.org/10.1007/s10462-018-09679-z

[10] P. Baier, S. Dragiev, *Challenges in Live Monitoring of Machine Learning Systems*, The Upper-Rhine Artificial Intelligence Symposium UR-AI 2021, ARTIFICIAL INTELLIGENCE - APPLICATION IN LIFE SCIENCES AND BEYOND, 2021.

DE LA DATE LA DECIZII: IMPORTANȚA MONITORIZĂRII SISTEMELOR ML IN APLICAȚII INDUSTRIALE

Rezumat: Sistemele de învățare automată (ML) au fost utilizate pe scară largă în diverse aplicații industriale, inclusiv optimizarea proceselor, mentenanța predictivă și controlul calității. Succesul sistemelor ML în aceste domenii depinde de capacitatea lor de a face predicții, decizii și recomandări precise. Având în vedere că toate modelele ML sunt în mod inerent perisabile, trebuie puse în aplicare măsuri de precauție pentru detectarea degradării de performanța înainte ca aceasta să producă rezultate negative. În această lucrare, explorăm importanța monitorizării sistemelor ML și diferitele zone care trebuie supravegheate. Prezentăm o implementare nouă, utilizând o combinație unică de componente pentru a obține o vedere de ansamblu asupra parametrilor de funcționare ai soluției ML, explorând use-case-ul al Process & Pipeline Services de detectare a fenomenelor de fisurare al pipeline-urilor datorate tensiunilor mecanice acumulate. In lucrare propunem elemente avansate de stricta noutate nemaiîntâlnite in literatura de specialitate.

Adrian-Ioan ARGESANU, PhD. Stud. Eng., Politehnica University Timisoara, Faculty of Automation and Computers, adrian.argesanu@student.upt.ro, Bulevardul Vasile Pârvan 2, 300223 Timișoara, Romania, Baker Hughes, Process & Pipeline Services, adrian.argesanu@bakerhughes.com, +4972447320, Lorenzstraße 10, 76297 Stutensee, Germany.

Gheorghe-Daniel ANDREESCU, PhD. Eng. Prof., Politehnica University Timisoara, Faculty of Automation and Computers, daniel.andreescu@upt.ro, +40256403507, Bulevardul Vasile Pârvan 2, 300223 Timișoara, Romania.