AISTech 2024 — Proceedings of the Iron & Steel Technology Conference
6–9 May 2024, Columbus, Ohio., USA
DOI: 10.33313/388/101

Automated, Scalable AI for Real-Time Monitoring of Steel Continuous Casting System

Joe Porter, Keval Bhanushali, Shreebhooshan B., Suhas Mehta, Nikunj Mehta

Falkonry Inc.
10020 N De Anza Blvd #200, Cupertino, CA 95014 USA
Phone:+1 408 461 9286
Emails: {joe.porter | keval.bhanushali | shreebhooshan.badrinarayan | suhas.mehta | nikunj.mehta }@falkonry.com

ABSTRACT

Smart steelmaking is impeded by the need for long, manual cycles of data collection, data preparation, domain analysis, model building, operationalization, maintenance, and root cause analysis. It is simply inconceivable that supervised approaches can be applied at any scale, given data quality issues that are prevalent in the industry. Furthermore, supervised approaches simply cannot surface behaviors that are not previously recorded and understood, which further limits their adoption. We have therefore looked for new solutions to achieve line scale Smart steelmaking primarily in the form of anomaly detection from SCADA and PLC data. In this paper, we demonstrate the automatability and scalability of two approaches - semi-supervised and self-supervised (hands-free) - with examples from continuous casting and demonstrate that the performance is comparable. A line-scale anomaly detection approach enables Smart real-time decisions over the whole steel mill. The hands-free approach also does not require any upfront investment in data quality or labeling as well as the need for ongoing manual model maintenance.

Keywords: Time Series AI, Self-supervised AI, Continuous casting, Pattern detection AI, Early warning system, Anomaly Detection, Condition-Based Maintenance

INTRODUCTION

In prior work, Falkonry has presented plant-level issues that must be addressed for a monitoring system to be considered scalable and 'hands-free' [4]. To recap and expand on that list:

- Plant data is messy: frequent gaps, signal noise, and irregular availability of measurements all complicate the construction, quality, and operation of high-quality monitoring systems.
- Plant conditions change frequently - changes in product grade, equipment, operational loads, policies, and many other factors all have a direct impact on the equipment performance, and therefore, measured signal data. As a result, predictive and early warning models demand a lot of ongoing maintenance to retain their accuracy.
- The data science process of collecting data, building models, hypothesis testing, and performance tuning is prohibitively costly at the scale of the whole steel mill and prone to human error.
- Knowledge of every possible failure mode *a priori* is impossible nor is it realistic to find accurate records of failures from the past. Supervised predictive models are a better version of the 'fail-and-fix' approach, but they still rely on time-consuming human understanding and observation. An ideal monitoring system should be able to reach beyond known behavior well into the unknown [8, 9].
- Rule-based alerts and modeling based on 'golden curves' are not scalable and are fragile [8,9]. Rule and behavior-based analytics require manual inspection of data and processes to divide nominal behavior from meaningful anomaly characteristics. As plant behavior changes these rules and curves must be revisited.
- To be useful, automated models must identify precursors to failures with sufficient time to address the issues, and with sufficient accuracy to avoid drowning in false positives. In the classical supervised modeling approach, this implies manual iterations to fine-tune models for higher precision while avoiding overfitting.
- Human attention does not scale. Existing automation systems increase the reach of human operators and managers, but the actual data coverage within the purview of human attention remains minimal. A full line-scale monitoring system must confer confidence that all system behavior is observed in a meaningful way that doesn't overwhelm the consumers of that data [9].

First, our semi-supervised learning approach and associated software tools (Falkonry Patterns) provide a level of automation that dramatically lowers the cost of predictive modeling [1] and removes the need for data science expertise to build and operate machine learning systems. Patterns Workbench (hereafter Patterns) presents the following features to help build robust classifiers:

1. Makes it easy to explore signal data visually, and to navigate and select signal data for automated learning and model-building
2. Makes labeling multivariate time series behavior easy, using a clustering and selection method
3. The clustering makes it possible to automatically create multivariate outlier detection in an unsupervised learning mode
4. Automatically builds classification models once data is selected and, optionally, labeled
5. Produces explanation of classification produced for any point in time along with its confidence in that classification
6. Makes it easy to activate classification models and supply real-time data to classification models
7. Records the classification results in real-time along with corresponding explanations and confidence level

Patterns achieves robust, flexible performance when compared with rule-based alerting, and reduces cost when compared with manual data science efforts. In many scenarios, we have demonstrated the effectiveness of our classification approach to achieve early detection of degraded system performance [1,2,5,7]. What the Patterns approach lacks is the ability to seamlessly scale automated predictive monitoring to all signals in a steel mill, and to adapt to the macro-level changes in line behavior. Patterns excel when users need to capture known behaviors and reliably report their occurrence. Patterns fall short in the face of massive scale, and novel/unknown anomalous behavior.

Our newer self-supervised learning approach and associated software system, Falkonry Insights (hereafter Insights), addresses the remaining problems of 1) scale and 2) the difficulty of characterizing unknown behaviors. With Insights, every signal in a steel mill has a corresponding sequence of predicted anomaly scores that indicate the level of novelty of each signal sequence. A key capability of Insights is the unique encoding of data windows to cleanly handle messy, noisy, and gappy data. The details of that process are beyond the scope of this article but have been previously presented [4]. Learning the normal behavior is automated for each signal, beginning once an adequate amount of data has been collected, and concluding within a matter of minutes. Self-supervised learning facilitates line-scale AI, further reduces the data preparation costs by eliminating the need for labeled data, and makes it much easier to address dynamic changes in the plant through rapid, incremental learning.

The continuous casting process, accounting for over 90% of global steel production, involves complex equipment spanning from the mold to multiple segments of casting rollers. Exposure to harsh environments such as high temperatures, high pressure, and oil pollution, further increases the complexity of the operating state of the caster at any point in time. Such a challenging environment amplifies the wear and fatigue of components, often leading to compromised production quality or even a complete unplanned shutdown of the caster. Failure modes are often coupled with each other, elongating the troubleshooting and resolution of the problem. Moreover, the degradation of caster components could occur at such a rapid rate that rule-based alert systems provide little to no time before the actual failure of the component. Therefore, the plant team must have an early warning system that can identify the unknown precursor conditions to failure (onset of degradation), so that timely diagnosis and repair actions can be scheduled based on the condition of the equipment.

TECHNICAL BACKGROUND

Semi-supervised learning approach (Patterns)

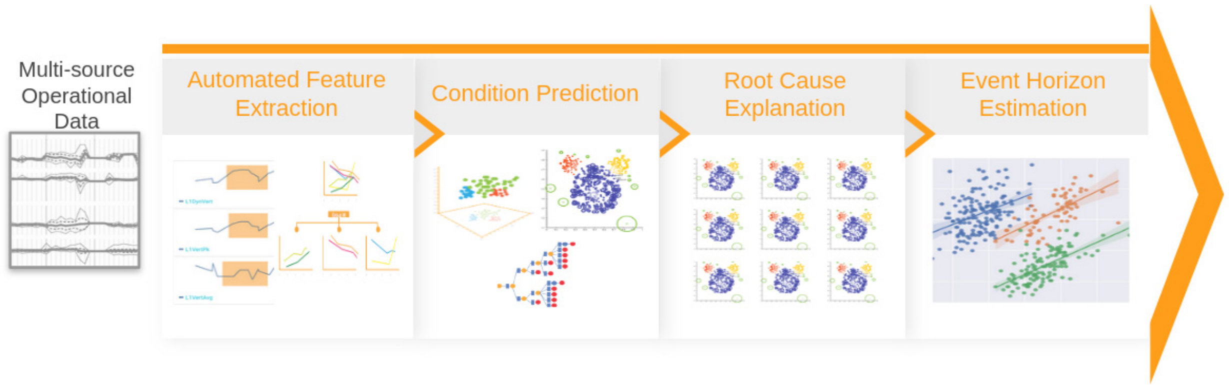

Figure 1. Falkonry Patterns semi-supervised data and ML pipeline [5]

In our semi-supervised approach [5] operational data is processed through four stages, as shown in Figure 1:

1. Automated feature extraction: This stage uses adaptive windowing to optimize the use of data history. The stage can take advantage of signal metadata provided by the user (such as data type or maximum allowed sampling interval) but does not require algorithm selection by the user. This design approach to automated feature extraction eliminates expensive data preparation efforts that are repeated for each application. This allows us to operate on gappy, noisy, and irregular data and maintain an efficient representation of input data without manual tuning.
2. Condition prediction: At this stage, we introduce labels to the data stream. Each time segment is assigned a label, whether manually provided or generated by the system from data clusters.
3. Explanation scoring: At this stage, we automatically quantify the contribution of each signal in the ensemble to the condition value provided above. Adjacent points are used for context.
4. Event horizon estimation: This stage estimates the time from the present until a predicted occurrence of a known target event. This calculation is dynamic and is meant to be updated as operational characteristics change. Many factors could be associated with the event horizon, and as such could affect the time estimate.

Self-Supervised Learning Approach (Insights)
Our self-supervised processing pipeline [3,4] takes a different approach in order to improve the scalability and autonomy of our solution.

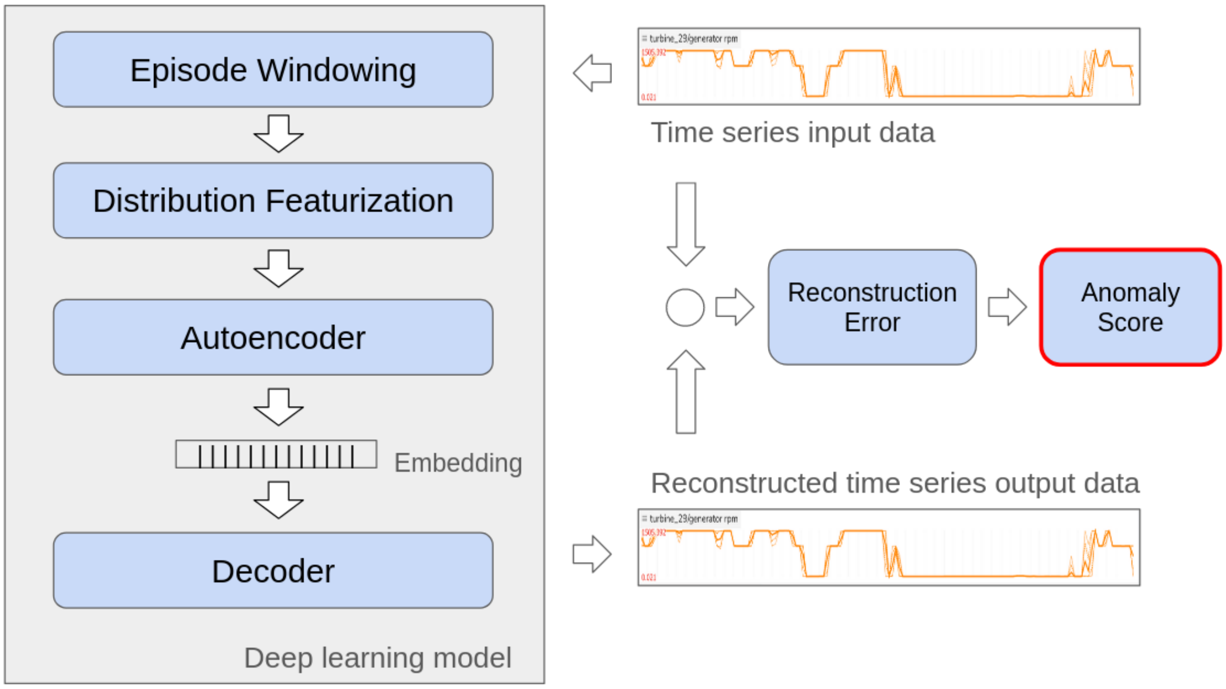

Figure 2. Falkonry Insights deep learning architecture

The self-supervised approach employs deep learning to create a fully automated system. Figure 2 shows the learning architecture for each signal. We use an autoencoder together with a signal reconstruction task to create a lightweight deep-learning model for each signal. Each signal's model learns to reconstruct signal waveforms from input samples that it has seen, effectively encoding nominal behavior into the model weights as it learns. If anomalous data is present during learning, it will be learned in proportion to the density of its occurrence. In this way we avoid having to select training regions manually - we simply start with operational data. If the data is imperfect, the model is robust enough to learn that odd things are rare if they occur rarely within the training set. If additional learning is required (i.e. sufficient learning on signal behavior has not occurred), we can determine that and schedule the model for an incremental learning update. If the wrong behavior has been learned (e.g., we happened to start during a maintenance cycle) then the inference performance of the model will indicate that it needs to be retrained from scratch. Data collection for self-supervised learning is much cheaper and easier than in the supervised case (totally 'hands-free'), and the efficiency of learning means that a model can be updated or even retrained from scratch in minutes.

Lightweight deep learning may seem like an oxymoron, but it's key to our approach. Each signal is monitored by a separate model. We often hear concerns about the effectiveness of this approach when contrasted with multivariate methods. In practice, because we are not directly modeling dynamics, single-signal monitoring works very well. If a single component fails, then all or most of the signals associated with that component will demonstrate high anomaly scores. These co-occurrences allow us to more easily isolate the location of an event within the mill.

Operationally, two key innovations are required to make the system scalable. First is a real-time automation infrastructure for managing signal data and operations of deep learning models. As a new signal is created, the automation infrastructure schedules the signal for self-supervised learning pending the accumulation of sufficient data. Once learning is complete, that signal is marked as ready for inference. New data collected on that signal is immediately eligible for anomaly scoring. The learning architecture is designed to support incremental improvement as well as to reset a model that is no longer aligned with the behavior of the signal being monitored. The ability to fully automate incremental learning is still under development as of this writing.

The second innovation is the requirement to aggregate anomaly information into human-digestible alerts and statuses. The self-supervised models do not require a priori knowledge of problematic behavior or isolating normal behavior, so they are also capable of surfacing previously unknown (but potentially problematic) behaviors. We have also developed mechanisms to consolidate anomaly scores into a single anomaly alert that aggregates information about anomalous behavior across multiple signals. Details of such anomaly alerts are beyond the scope of this paper but can be obtained from our publicly available repositories [8,9].

COMPARATIVE STUDY

To understand and compare the effectiveness of semi-supervised and self-supervised learning, both approaches were applied to the continuous casting process of steelmaking. This process consists of mold casting, oscillators, and a fixed number of segments that slowly drag the slab or billet and rotate it from a vertical to a horizontal axis. Each piece of equipment is highly instrumented. We utilized around 800 PLC tags or signals that generated readings at a rate of 10Hz from this line. Out of these 800 signals, 660 of them monitored different parameters of the operational process while the remaining 140 served as metadata for the ongoing operational process.

Semi-Supervised Learning Approach (Patterns)

A. Definition

For purposes of line-scale event detection and early warning, we used the semi-supervised approach for learning various recurring conditions as well as for identifying outliers. To do so, we selectively supply only those periods of time when normal conditions prevail. This required additional setup effort in identifying the time periods when the caster was under normal operations. Consequently, any new behavior that is not present in the inputs learned by the model will be reported as the “unknown” condition. This eliminates the need to provide any labels at all.

Assets within the continuous casting process had more than one type of downtime event, where each event may happen once or at most twice in a year. This resulted in defining around 100 use cases and relevant signals were selected for each use case. Across these 100 use cases, additional documentation was collected such as maintenance schedule, different operational states, knowledge of previous events and the appropriate validation time range. The definition process took around 3-5 weeks.

B. Setup

Once the use case definitions were complete, the model setup for those use cases began. This involved narrowing down to the right set of signals and selecting two main hyperparameters - window bounds and generalization factor. The first few use cases took longer due to the uncertainty of which signals were relevant to that event. Eventually, we developed a recipe to speed up the model setup. By removing the criteria for a perfect set of signals for each use case and eliminating the need to experiment with hyperparameters for their model, the majority of use cases had completed models within 4 weeks.

C. Validation

For a model to be deployed in production, it had to isolate the noise as much as possible while detecting the correct instances of early warning and event behaviors. The criteria for a condition to be considered as early warning was time-bound - ranging from a few hours to several days depending on the asset and the failure mode. Besides, the distribution of the condition in the validation range determined whether it was under an acceptable limit or not. For this line, any given asset was averaging to operate normally 95% of the time. Once the criteria were satisfied the model could be deployed to monitor the use case in near real-time interactively. It was observed that 85% of the models satisfied the validation criteria.

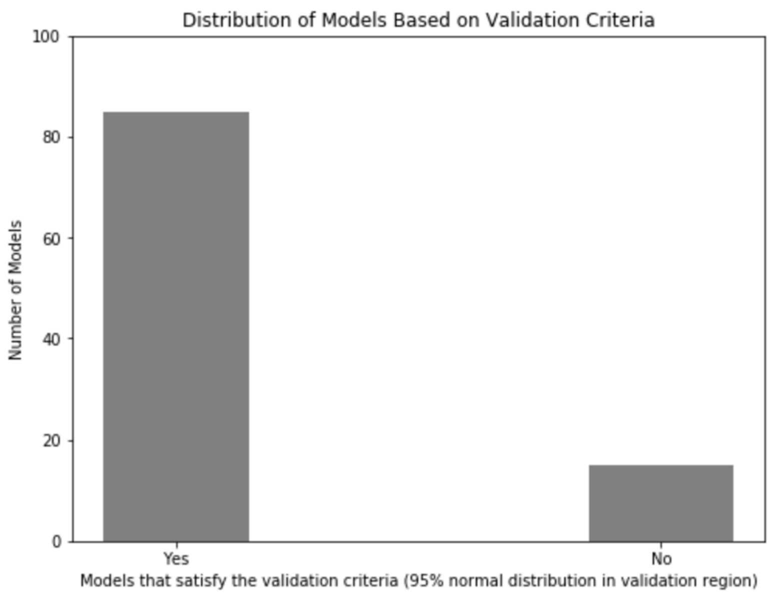

Figure 3. Distribution of unknown conditions across 100+ use cases

D. Summary of Results

From our earlier paper [5], Table 1 shows the preparation and effort required to model specific precursors of failure, i.e., early warning, in different use cases across the continuous casting process. Each of these use cases took approximately two weeks to develop, "with one person for data selection and connectivity and two part-time users supporting ML application development and validation." The entire collection of 50 use cases took 18 months to develop.

Compared with this, we were able to set up, develop, and deploy 100 use cases within 9 weeks. Additionally, as Figure 3 shows, the distribution of unknown behavior across these use cases was within reasonable limits as 85 use cases showed acceptable levels of unknown behavior. This study does not cover the periodic maintenance of models required to address operational changes such as changes in components, process parameters, etc. Additional time and resource investment would be required to maintain the model quality for the given validation criteria.

Asset Type	Application	Data Considered	Data Used	ML Approach	Application Development Time
Mold oscillator	Oscillator health	130 parameters 135GB	1 month, 29 parameters	Unsupervised, Semi-Supervised	20 iterations over 12 hours
Bending rolls	Precursors to jammed rolls	30 parameters 100GB	7 months, 9 parameters	Semi-Supervised	20 iterations over 12 hours
Dummy bar	Precursors to failure	21 parameters 68GB	6 months, 8 parameters	Semi-Supervised	20 iterations over 6 hours
Straightener	Precursors to jammed rolls	17 parameters 10GB	2 months, 15 parameters	Semi-Supervised	10 iterations over 10 hours
Segment roller	Precursors to jammed rolls	93 parameters 360GB	5 months, 9 parameters	Semi-Supervised	35 iterations over 40 hours

Segment roller	Segment bulging		4 months, 15 parameters	Semi-Supervised	20 iterations over 30 hours
Pinch roller	Precursors to failure	40 parameters 90GB	8 months, 8 parameters	Unsupervised, Semi-Supervised	40 iterations over 50 hours
Shears	Motor health	5 parameters 240 GB	1 week, 1 parameter	Unsupervised, Semi-Supervised	5 iterations over 5 hours

Table 1. Information on select use cases for the continuous caster using Patterns

Self-Supervised Learning Approach (Insights)

A. Definition

We used the existing definitions of use cases from the semi-supervised learning approach to group signals for purposes of visualizing anomaly detection results. Note that this grouping has no implications on learning whereas the semi-supervised learning approach bakes these signal relations into models. This makes the self-supervised approach more robust to engineering changes and communication issues. For example, even if some of the signals in a group experience interruption, the rest of the signals in the group continue to be considered for anomalies.

B. Learning

In contrast to the semi-supervised approach, the self-supervised approach (Insights) requires no setup effort. Insights is ready to learn the normal behavior of each signal, provided enough data is available in that signal to learn the signal behavior. Insights automatically waits for data to flow through data connections until the required data volume is received. In the table below, we can observe the learning latency of Insights for signals with different input frequencies, which indicates how much data is required for each of those sampling rates. Typical learning time for a single signal at 1 Hz is on the order of an hour, so initial learning and incremental learning are quick. The initial learning latency values in the table below represent the current minimum requirements to create a valid model for a signal. Some signals may need additional data in order to capture the majority of the variations present in its behavior. The inference latency represents the size of the time window being used for inference at that scale, so it represents the delay until the system produces the first full output once the normal behavior of that signal is learned. Inference results are current - there is no delay once they start to produce output. Clearly, slower signals have much longer starting latencies. Operational considerations such as approvals and validation may take additional time but the system latency from first data to first inference is minimal and far smaller than conventional approaches.

Sampling frequency (resolution)	Initial learning latency (hours)	Initial Inference latency (hours)
10 Hz (100 ms)	.14	.14
1 Hz (1 s)	1.4	1.4
.1 Hz (10 s)	2.9	2.9
17 mHz (60 s)	8.7	8.7
1.7 mHz (600 s)	45.3	40.8

Table 2. Minimum data required for learning at different time resolutions (Insights)

C. Validation & Summary

Insights follows a process of active learning, which means that it periodically conducts incremental learning and takes into account new data as it accumulates. As Insights focuses on detecting anomalies, the output for a signal is determined as useful based on the limited variation and severity of the anomaly observed for a given period of time. Assuming Insights has learned all the behaviors for a signal, the expected anomaly score level in a 2-week time frame should have low values most of the time. As the anomaly score of less than 4 is considered as a very low to no anomaly, it was observed that 90% of signals satisfied this criterion within 5 weeks of learning. This implies that the self-supervised system has sufficiently learned the normal signal behavior and is ready to report anomalies in continuous caster within only about one month of activation.

In Figure 4, we observe the distribution of anomaly scores over January 2024. The signals being monitored for anomalous behavior are the process signals in a continuous casting line. It is observed that 592 signals satisfy the validation criteria out of a total of 660 signals.

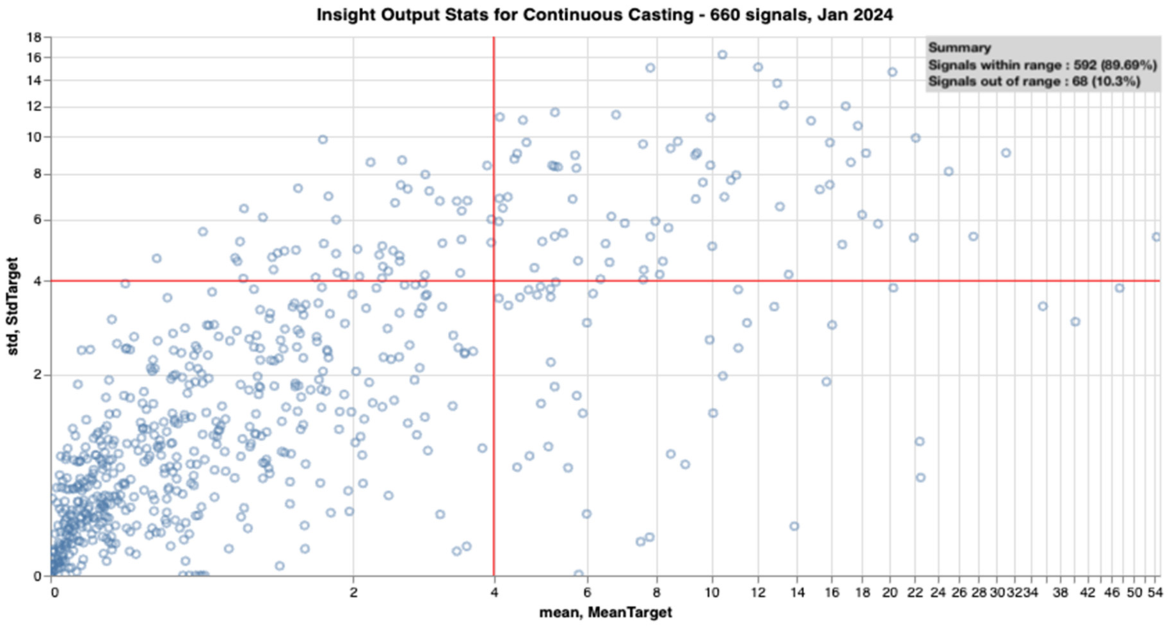

Figure 4. Distribution of anomaly score value for Continuous casting line

DEMONSTRATION CASES & RESULTS

Case 1: Detecting a 'Known' Event in the Caster Segments

A few months after the deployment of semi-supervised Patterns and self-supervised Insights on the continuous casting line, a pinch roll failure occurred on a segment resulting in downtime. This is a known event and was identified as one of the primary use cases to be monitored using Patterns and to potentially detect any early indicators before the pinch roll failed.

Observing the Patterns output, it identified an unknown condition two days prior to the pinch roll failure event. This observation was validated by the continuous casting experts who determined that the measured torque exhibited an inconsistent behavior in the time frame highlighted by Patterns. Figure 5 shows the waveforms of the signals involved in the pinch roll line (signal traces in blue at the bottom), and a representation of the Patterns output in the traces at the top. As shown in Figure 5, precursor behaviors were detected twice in the days prior to the actual failure. The red sections in the green band at the top show these precursor indicators to the pinch roll failure. We are reminded that the Patterns model was built by manually selecting appropriate data and is a labor-intensive process.

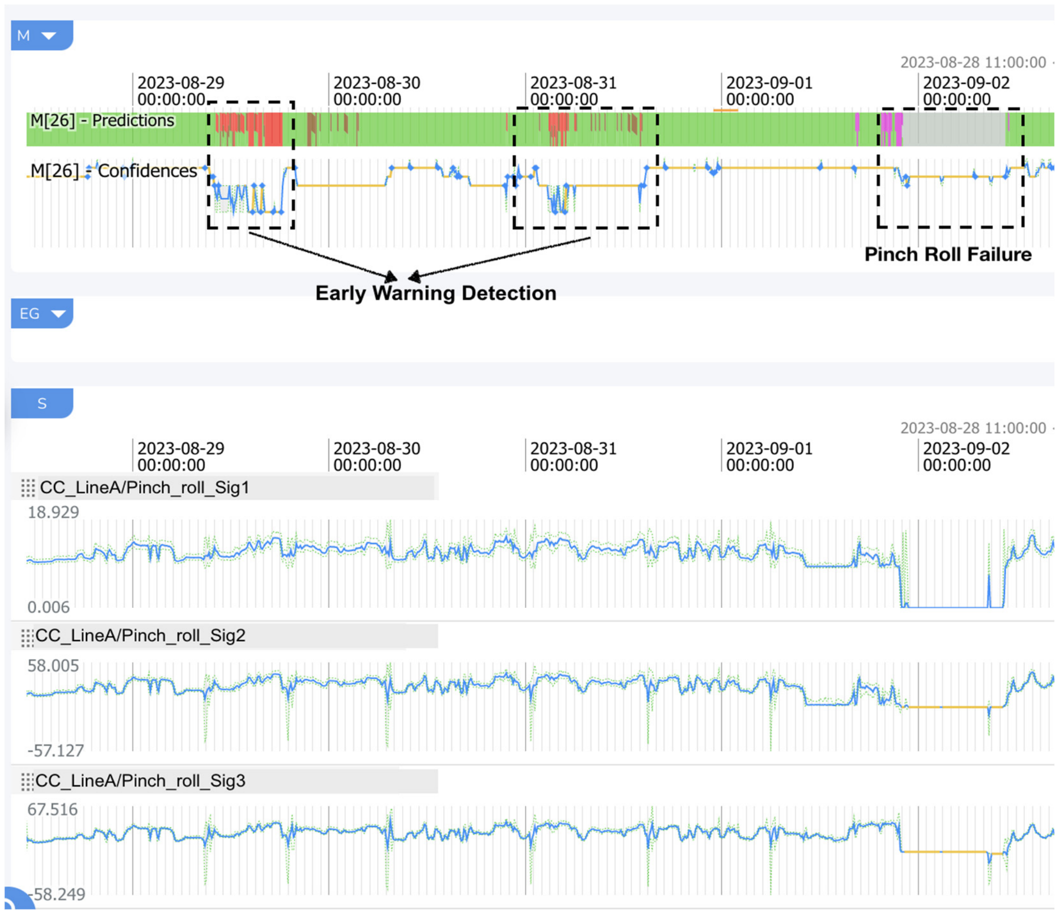

Figure 5. Patterns detecting an early warning for a known event

The Insights output for the same time frame in the same equipment provides additional context to the failure in the form of anomaly levels. While Insights identified the same regions observed in Patterns as highly anomalous, it identified additional anomalies as a result of short-duration torque spikes that occurred without any changes in the control signals. Besides the torque signals, there were 9 more signals that were flagged as anomalous by Insights belonging to a specific side of the segment, thus helping the operations potentially identify where the issue originated. Figure 6 shows the traces for torque signals in the pinch roll chain. The waveforms are shown in blue, and the purple band beneath each waveform represents the anomaly level. Yellow bands indicate high anomaly values for unexpected behavior. They were the same as unknown behaviors detected by Patterns. The advantage of Insights is that it did not require any specific guidance of normal/nominal behavior. Further, without a priori knowledge, Insights was able to help isolate the probable causes of the failure outside the signals that are immediately involved in this use case.

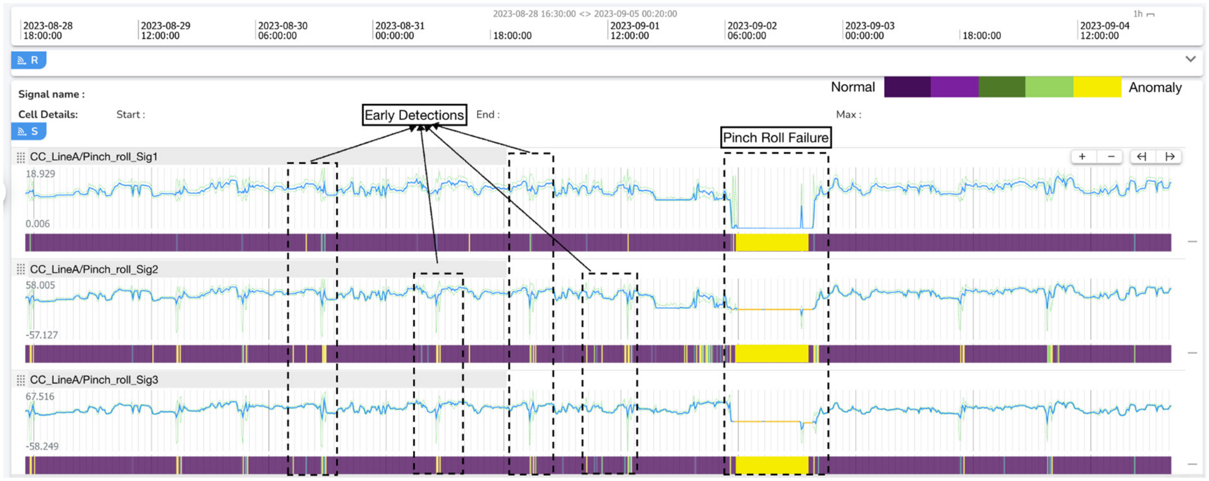

Figure 6. Insights detected multiple early warnings & the known event

Case 2: Detecting a Novel Event in the Caster Mold

In this case, an event was observed in an asset during the molding process of continuous casting, leading to a brief shutdown in the continuous casting process.

We observe that Patterns was able to identify the unknown precursor conditions a few hours before the caster shutdown. These conditions on the signals served as early indicators. Patterns also identified that the event re-occurred a couple of hours after the asset was restarted. However, it did not identify what led to the problem after the asset was restarted.

Figure 7 shows the signal waveforms involved in monitoring the asset for the molding process (signal traces in blue at the bottom, signal names are anonymized), and a representation of the Patterns output in the traces at the top. As highlighted in the figure the red bands seen are the unknown bands that indicate the 'early detections' and the 'known detections'.

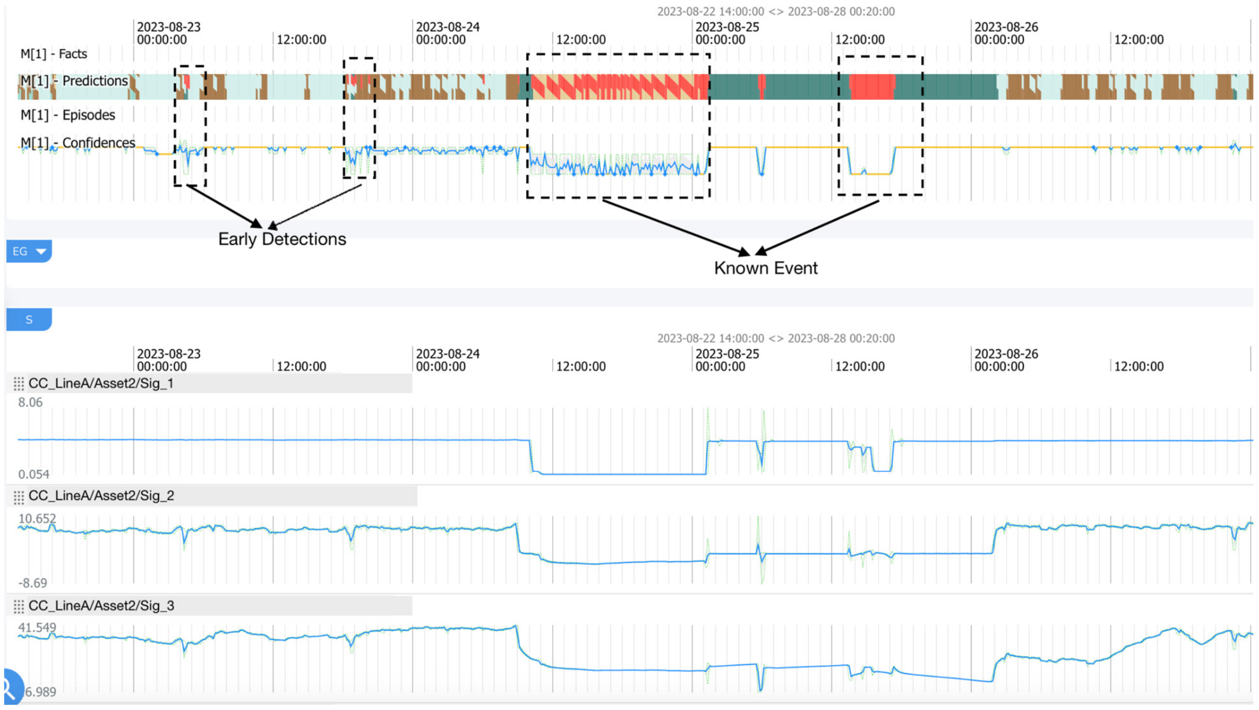

Figure 7. Patterns Detecting a known event and early warning indicators

With Insights, we observe that the precursor conditions on the signals are detected ahead of the failure, and represented by high anomaly scores. Also, as the operations are restarted after the first occurrence of the event, Insights surfaces anomalies in 'CC_LineA/Asset2/Sig_3'. This additional information about the novel behavior points us to note the restart issue faced by the component. Insights points us to precise causal factors, leading to the instant root cause of the issue after restarting the operations and eliminating the manual troubleshooting time, and potentially leading to a shorter downtime period. Again, Insights did not have the advantage of supervised guidance to create a functioning model - the results come solely from the machine learning approach and the automated learning and inference system that supports it.

Figure 8 shows the waveforms in blue color, which are traces for a subset of signals in this asset. The band beneath each waveform represents the anomaly level which has colors ranging from deep blue to yellow that indicates the anomaly level. It is observed that Insights observes anomalous behavior across signals before the occurrence of the first event. Without a priori knowledge, Insights identified additional early indicators as well as the root cause for an issue after restarting the operations on the asset.

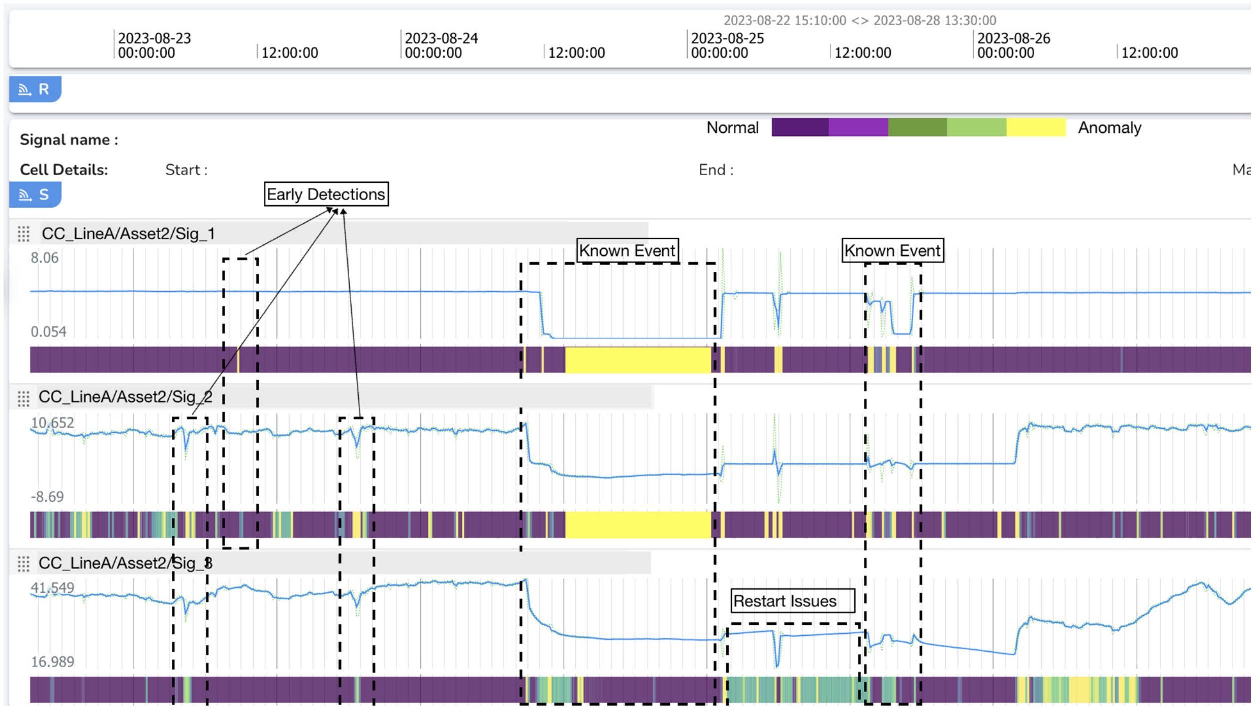

Figure 8. Insights detects early warning anomalies for the same signals, also detects restart issues

CONCLUSION

Deployment of the Falkonry Insights dramatically reduced setup effort for locating early warnings in the continuous casting process. The labor-intensive semi-supervised approach took 60% more wall clock time than the self-supervised approach. Falkonry Insights also removes the need to manually and repeatedly perform model maintenance compared with the Falkonry Patterns approach. With the help of two examples, we have shown that the self-supervised approach identifies the same behaviors that the semi-supervised approach has indicated. The self-supervised approach gives a better understanding of root causes and doesn't limit the analysis to a predetermined set of signals. We expect that the self-supervised approach will provide sufficient and more time for preventive action than the semi-supervised approach, which will be presented in our future work. The computational scaling of our lightweight deep learning, Falkonry Insights makes a self-supervised approach economically viable for real-time monitoring at line-scale. Self-supervised machine learning, together with continuous and automated improvements from live data and anomaly event aggregation make Smart steelmaking a practical and immediate possibility.

REFERENCES

1. C. Lee, "Falkonry on Intelligence - Discovering New Patterns in the Noise", Falkonry, Inc., 5 Aug 2020. [Online]. Available: https://falkonry.com/blog/falkonry-on-intelligence-discovering-new-patterns-in-the-noise

2. C. Waters, R. Talla, and K. Bhanushali, "Time Series AI for Anomaly Detection and Diagnosis in Steel Production" in AISTech 2022 - Proceedings of the Iron & Steel Technology Conference. Pittsburgh, PA.

3. N. Mehta, C. Singh, V. Toroman, and D. Kearns, "Automated, fast multi-timescale, time series anomaly detection for industrial data with Time Series AI" in Advanced Process Control Smart Manufacturing of Semi.org, Abstract #22025, 2022

4. J. Porter, K. Bhanushali, D. Kearns, N. Mehta. "Hands-Free" Fully Autonomous, Plant-Scale, Anomaly Detection AI. AISTech 2023 - Proceedings of the Iron & Steel Technology Conference. Detroit, MI.

5. C. Waters, B. Klemme, R. Talla, P. Jain, N. Mehta. Transforming Metal Production by Maximizing Revenue Generation with Operational AI. AISTech 2021 - Proceedings of the Iron & Steel Technology Conference. Nashville, TN.

6. E. LaMalfa and G. LaMalfa, "Unsupervised Anomaly Detection in Time Series with Convolutional-VAE" in Journal of Mathematics and Statistical Science, pp 103-108, 2020.

7. N. Mehta, "Event Horizon Estimation - Time to Any Critical Event," Falkonry Inc., 17 November 2020. [Online]. Available: https://falkonry.com/blog/event-horizon-estimation-time-to-any-critical-event.
8. N. Mehta. Advance your knowledge horizon with AI-based anomaly detection. Falkonry, Inc., 10 August 2023. [Online]. Available: https://falkonry.com/blog/advance-your-knowledge-horizon-with-ai-based-anomaly-detection/
9. S. Parwatay. Deep dive into Falkonry's unique tech capabilities with CTO Dan Kearns. Falkonry, Inc. 25 August 2022. [Online]. Available: https://falkonry.com/blog/deep-dive-into-falkonrys-unique-tech-capabilities-with-cto-dan-kearns/