SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

AI for IT Operations (AIOps)

Using AI/ML for Improving IT Operations

A Technical Paper prepared for SCTE by

Hongcheng Wang
Distinguished Engineer, Machine Learning
Applied AI & Discovery, Comcast
1325 G Street NW, Washington, DC 20005
332-301-5055
Hongcheng_Wang@comcast.com

Praveen Manoharan, Applied AI & Discovery, Comcast

Nilesh Nayan, Applied AI & Discovery, Comcast

Aravindakumar Venugopalan, Applied AI & Discovery, Comcast

Abhijeet Mulye, Applied AI & Discovery, Comcast

Tianwen Chen, Applied AI & Discovery, Comcast

Mateja Putic, Applied AI & Discovery, Comcast

Table of Contents

Title **Page Number**

1. Introduction 4
1.1. What is AIOps? 5
1.2. The Difference from MLOps 5
1.3. Why AIOps? 5
2. AIOps Platform Architecture 6
2.1. Real-Time Data Processing 7
2.2. Scalable Architecture 7
2.3. Data Storage and Retrieval 7
2.4. Real-Time model training and model packaging 7
3. AIOps Use Cases 8
3.1. Proactive Monitoring 8
3.2. Smart Alerting 8
3.3. Topology Analytics and Root Cause Analysis (RCA) 8
3.4. Log and Trace Analytics 9
3.5. Cohort Analysis 9
3.6. Automated Remediation 9
3.7. Smart Orchestration 9
3.8. Release Management 9
4. Anomaly Detection 10
4.1. Anomaly Detection Algorithms 10
4.2. Anomaly Detection Platform 11
5. AIOps – Operators' Perspective 12
5.1. Onboarding a tenant 13
5.2. Training a metric with model 13
5.3. Schedule a metric for real-time inference and prediction 13
5.4. Configure Alerts and RCA 13
6. AIOps Case Study: Connected Living Object Detection Operation 13
6.1. Connected Living Object Detection Operation 14
6.2. Log mining and Correlation Analysis 16
7. AIOps Case Study: Rex Browse and Search Operation 19
7.1. Dependency Graph 19
7.2. Root Cause Analysis with Dependency Graph 20
7.3. Failure Prevention and Auto Remediation 22
8. AIOps Case Study: Release Management for RDK Firmware 24
8.1. Machine Learning Model 24
8.2. Model Results 24
9. Conclusions 25
10. Acknowledgements 25

Abbreviations 26

Bibliography & References 27

List of Figures

Title **Page Number**

Figure 1 – AIOps High Level Architecture Diagram 6
Figure 2 – Anomaly Detection Platform 12
Figure 3 – Metric in Object Detection Operation 14
Figure 4 – Anomaly Detection Predictions in Object Detection Operation 15
Figure 5 – Manual Log RCA in Object Detection Operation 17
Figure 6 – Automatic Log RCA in Object Detection Operation 18
Figure 7 – Manual Flow of Issue Root Cause Diagnosis by Rex Operations Team 19
Figure 8 – Dependency Graph Created in Rex Case Study 20
Figure 9 – AIOps Showing Automatic RCA Correlated Ranked List 21
Figure 10 – AIOps Showing Probable Root Cause Metric Time-series 22
Figure 11 – Percentage of Pods Having GC activity per 5-min Period 23
Figure 12 – Instances Showing Abnormal GC Count Over Time 23
Figure 13 – Precision Recall Curve for the model (Left FIgure); 5G WiFi Signal Distribution for Old Firmware Version and New Version (Right Figure) 25

1. Introduction

The proliferation of microservices as a dominant IT architecture has created opportunities as well as challenges for operations teams responsible for maintaining software reliability. When production systems deviate from service-level objectives, operations teams must detect failures and discover their root causes to promptly resolve the issue. These teams are most often focused on minimizing mean time to resolution (MTTR) or mean time between failures (MTBF). The process of identifying, diagnosing, and resolving issues in cloud microservice architectures largely falls into two phases: anomaly detection (AD) and root cause analysis (RCA).

AD is the process of identifying anomalies that correspond to system failures. RCA is the process of determining the reason why an anomaly occurred and identifying the originating service or system. Anomalies are typically detected by defining margins of normal operation on key performance indicators (KPIs) and setting alerting thresholds that generate notifications. RCA is then typically performed by inspecting the system that generated the alert and tracing the problem back to its source. Operations teams use log, trace, or metric data sources, often displayed in dashboards to diagnose and debug problems.

However, there are several problems with this and related existing approaches:

(1) ***Simplistic AD still dominates:*** State-of-the-art failure detection is still based on simple thresholding, which is prone to drift, and fails to capture low-frequency events such as weekends, holidays, or special events. Failure detection based on simple thresholding, misses opportunities to preemptively diagnose problems, thereby increasing MTTR.

(2) ***Manual RCA still dominates:*** Root cause analysis is the most time-consuming step of issue resolution, largely due to a reliance on a human in the loop. When an alert is received, reliability engineers spend significant time identifying the root cause by looking at numerous plots, traces, and logs. This work is repetitive, tedious, and ripe for automation.

(3) ***Excessive alert volume:*** Operations teams often receive a large volume of alerts, many of which are false or redundant. These are generated by rules that often remain unchanged for the life of the application. Further, the volume of services in a microservice application makes it difficult to know if a service has failed on its own or as part of a cascade.

(4) ***New deployment challenges:*** When a new product or service is deployed, the operations team keeps a closer eye on the alerts, metrics, and system performance. The decision to move forward to 100% general availability (GA) or roll back to a previous version usually takes unnecessary lead time, which may create negative customer impact.

(5) ***Institutional knowledge monopolies:*** Often the knowledge needed to quickly debug operational problems is held by a small number of individuals on the team. Root cause analysis can be time-inefficient except for the few individuals who hold a monopoly on that knowledge.

According to the report by Smartsheet [1], nearly 70% of employees say that automation reduces the time wasted on repetitive work, and out of which nearly 60% believe that if repetitive jobs were automated, they would save 6 or more hours (almost a full workday) each week. This brings huge opportunities for AIOps.

1.1. What is AIOps?

AIOps (AI for IT Operations) uses artificial intelligence and machine learning (AI/ML) and big data analytics to identify or predict IT operations issues in a timely manner and help the DevOps team quickly identify the root cause of the issues. A machine learning model can learn the conditions that lead to an alert and can predict when an alert is about to occur. When an alert does happen, ML based RCA is used to generate candidates of sources of failure to be diagnosed by a human operator. Thus, it can reduce MTTR by automating repetitive jobs, present failure, and provide better decision making.

AIOps has recently received extensive attention from industries and academia. According to a survey by Reportlinker – "AIOps Platform Market Forecast to 2028 - COVID-19 Impact and Global Analysis by Component, Deployment, Organization Size, and Vertical" [2] , the AIOps platform market size is expected to grow from $ 2.8 billion in 2021 to $ 19.9 billion by 2028. It is estimated to grow at a compound annual growth rate (CAGR) of 32.2% from 2021 to 2028. Apart from the market growth, its impact can save much more than that by providing the predictive abilities which will lead to efficient utilization of resources so that companies are able to cut additional costs related to overprovisioning of their cloud and other resources. It will also help in better application maintenance resulting in a better overall customer experience, thus positively contributing to business revenues.

1.2. The Difference from MLOps

The terms "AI" and "ML" are interchangeable in many contexts. In this context, however, the meaning of MLOps and AIOps are significantly different. MLOps refers to machine learning model operations, from data acquisition to model development, testing, validation, and deployment. MLOps seeks to increase automation and improve the quality of production ML models, by focusing on the operation of ML models, and leveraging the continuous integration/development (CI/CD) practice of DevOps in the software field.

The AIOps methodology is applicable to any IT operations, including MLOps. AIOps could make ML model operations more reliable and cost effective. On the other hand, the MLOps pipeline could be leveraged to make the operation of AIOps itself more efficient and reliable.

1.3. Why AIOps?

AIOps aggregates data from multiple sources and provides context and insights when problems occur. It improves the visibility (observability), reliability, availability, and cost of IT operations. In general, AIOps provides the following key business benefits:

(1) ***Improved system availability:*** AIOps improves availability by reducing MTTR in several ways. First, predictive AD can intelligently identify issues before they occur, automatically categorizing issue criticality, and preventing unnecessary escalation of issues. Second, automated RCA significantly narrows the scope of the problem on which a human operator must focus, greatly reducing the amount of time needed to reach resolution. Third, capturing institutional knowledge to a model means faster issue resolution, even if less experienced operators are on call.

(2) ***Reduced operational cost:*** There are multiple ways AIOps reduces operational cost. First, as AIOps sends out fewer alerts and automates RCA, the resulting reduction in workload could potentially reduce the headcount of operations teams. Second, as ML models can predict the pattern of traffic from historical data, AIOps can help to orchestrate resources more intelligently for cost savings. Third, AIOps helps to quickly identify any potential issues within limited

deployments and provides better insights and decision making before promotion to GA, reducing unnecessary use of valuable human and computing resources.

(3) ***Improved employee experience:*** The reduced number of false alarms creates more efficient (noise free) work for reliability engineers while lowering the overall work volume. Online learning models eliminate threshold drift and reduce manual effort. The AIOps system can act as a partner in a pair-debugging strategy, that enhances the capabilities of the human operator. The overall reduction in issue volume results in a happier, more productive operations team by eliminating pager fatigue, allowing them to focus on more meaningful tasks.

Our AIOps team is striving to help address the operational challenges with AIOps. We have explored and experimented on several use cases including:

(1) ***Intelligent Infrastructure Monitoring (IIM):*** We built the state-of-the-art anomaly detection technology to alert the DevOps team with detected anomalies based on load and resource utilization to prevent application failure as early as possible.

(2) ***Root Cause Analysis (RCA):*** When there is an anomaly or a failure in the operation, we correlate the system and application log data with the detected anomaly and help the team to quickly identify the root cause and recommend the correct actions to the team.

(3) ***Release Management (RM):*** We use ML to help manage the release by quickly identifying when a gap occurs with a partial rollout.

In this paper, we will give an overview of AIOps use cases and summarize our findings from several practical case studies from IoT (Internet of Things), content discovery, and RDK (Reference Design Kit) applications.

2. AIOps Platform Architecture

In this section, we describe the high-level AIOps platform architecture we built, as shown in Figure 1.

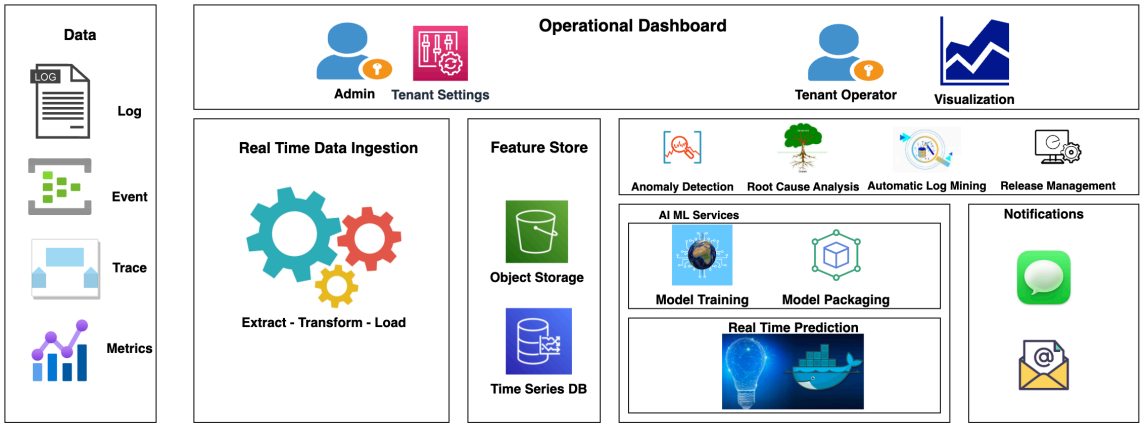

Figure 1 – AIOps High Level Architecture Diagram

AIOps architecture address end to end solution right from data ingestion to data transformation, data storage, model training, real-time prediction, fine tuning of model and notifications.
The data layer is composed of the data from different tools (e.g., metrics and log data sources) and systems, internal or external. This data may include a large scale of logs, events, traces, and

metrics data from applications onboarded onto the AIOps platform. Data transformation layer performs required transformation on the data and load to Feature Store. The data is ingested into the backend, and further engineered (e.g., transformation, aggregation) to store in the feature store (Object Storage or Time Series Database). AI/ML Service is the core of AIOps that enables model training, model tuning and real time inference for any selected operational metric from the feature store. Model training layer gives the ability to an operator to select, and tune required model to trial run and fine tune until the operator satisfies with required precision, recall against operational metrics and save the configuration for real time prediction. Model training can be supervised, semi-supervised, or unsupervised. The real time prediction layer performs prediction on the scheduled frequency against the scheduled time and stores the results. High level use cases like Anomaly prediction, Root cause analysis, automatic log mining, release management is performed. Notification layer alerts users on the configured channel like messenger , email etc. The data can also be visualized via an operational dashboard. The dashboard also provides a visualization of the output generated from the AI/ML engine and allows operators to provide feedback which can be incorporated in the models for better AI/ML predictions. With some business logic or high-level rules, we also allow generation of a report, which will alert the operations team via e-mail, SMS, or messenger. The major challenges in building AIOps architecture are listed in the following subsections.

2.1. Real-Time Data Processing

Real-time data processing is critical for AIOps, as timely prediction of application issues and outages brings strong value to the operators. This involves designing suitable architecture that can handle complex tasks including data ingestion, model prediction, and alerting in real time.

2.2. Scalable Architecture

Scalability is another key aspect in AIOps since the architecture must handle thousands of metrics coming from different application services, ingesting real-time metrics every second, and inferencing anomalies at seconds' level. For example, for a scale of one thousand metrics that are ingested every minute and inference also made at minute level, we are at looking at more than 1 million model inferences per day.

2.3. Data Storage and Retrieval

As data is the key for any ML tool, access to historical data will help the ML model train over a longer duration which can lead to a better model fit and reduced bias. AIOps architecture facilitates data storage and retrieval for longer durations with the ability to retrieve data with minimum latency.

2.4. Real-Time model training and model packaging

With AIOps, operators can experiment with multiple machine learning models, train a model for a specific metric with historical data, package the model for inference instantly, and schedule the model for real-time inference on that metric. Operators can also provide feedback on the test inference and fine tune the model in real time. So, real-time model training and packaging is one of the core requirements fulfilled in our AIOps architecture.

3. AIOps Use Cases

Through AIOps, we are aiming to empower DevOps teams to perform their tasks efficiently. The following are the major use cases which we plan to incorporate in the AIOps platform for the benefit of operators:

3.1. Proactive Monitoring

Operation teams have been using monitoring system and observability platforms to configure alerts for their applications. The drawback of such an arrangement is that the alert threshold remains static, whereas the applications, or their usage, evolve over time due to various reasons requiring operations teams to regularly adjust their alert thresholds to avoid noise caused by false alarms. Also, there can be cases where certain problems in the system go uncaptured.
For example, batch jobs happening at a specific time can create a spike in application metrics. Spikes during that time would not be an anomaly since it's an expected behavior but at other times it could be an anomaly. Perhaps the batch job is taking a longer time than expected and if so, this is an anomaly to notify. In a traditional setup, this could be missed since thresholds are static and don't take into account such changes in data behavior.
AIOps learns from historical data. Especially in this case, the model would learn from the time-series data pattern and seasonality trends to capture these abnormal behavior and notify accordingly.

3.2. Smart Alerting

Alerting in conventional systems is based on static limits. In AIOps, we take advantage of AI to have dynamic thresholds depending on the trend of data. Apart from the metric threshold rules, we can also have rules based on model confidence. Confidence of model can be described as the certainty or strength in prediction done by a model on certain data, whether it is anomalous or not. This confidence is developed by the model learning from historical data over a period of time. This can even surpass human performance since sometimes it's not possible for the operators to continuously monitor such high loads of data manually, whereas ML models can do that easily.
This feature is useful in predicting potential system latency or downtime which normally wouldn't be captured using metric threshold-based rules. Thus, this feature helps DevOps with such cases and in turn, improves the customer experience with our applications.

3.3. Topology Analytics and Root Cause Analysis (RCA)

Topology analytics is used to establish a dependency topological graph of application, network, and infrastructure for a complex system, and to drill down to the root cause of the issue. Since AIOps is ingesting live metric and log data, AI models can predict anomalies in real time. These predictions are further used to correlate anomalies between various system metrics using statistical techniques such as Pearson Correlation Coefficient [6] , and give a list of most probable root causes. It will be a much faster process compared to the operator doing RCA manually over thousands of metrics at a time. This feature will immensely help the DevOps team in automatic identification of problems and help them reduce MTTR in general.

3.4. Log and Trace Analytics

Each application may also have an immense volume of application-specific log and trace data. By leveraging Natural Language Processing (NLP) tools, we can extract the log metrics and their context from certain error/warning entities present in such data and perform anomaly detection against it. Additionally, we can also do correlation analysis to correlate the log anomaly with application metrics anomaly which can help the operations team to quickly identify the root cause of the issue. Similar analysis can also be done for trace data.

3.5. Cohort Analysis

Building on top of RCA, cohort analysis can provide further insights into system performance. It can point the team towards the probable issue in particular parts of their system even though they haven't received any specific alerts on them.
Multi-variate time-series analysis, i.e., time-series analysis done simultaneously on multiple metrics and clustering on anomalies as well as error logs, can correlate and identify such groups of metrics or systems that are causing problems, and can notify the team accordingly in advance for better maintenance of their application, all leading to better customer experience.

3.6. Automated Remediation

Consider a case where an application is deployed in a Kubernetes cluster, a group of nodes used in a Kubernetes deployment. Assume there is a 'garbage collection' system metric emitted from the cluster pods indicating one of these: application might go to a bad state or there's a problem with the pod itself. In general, this metric count is ignored since it's a normal behavior of the application. In some cases, though, it could actually be a problem with the cluster pods. Time-series models can detect such changes in data trend patterns and provide alerts to the team. Going one step further, the system can automatically take some actions (e.g., auto-restart) for remediation before it results in any application downtime or customer impact.

3.7. Smart Orchestration

Many teams use major cloud providers' auto-scaling features or their own customized rule-based scaling for their application resources. Through AIOps, we provide them with more intelligent auto-scaling abilities. Since the model will be able to forecast demands by learning the pattern and behavior through historical data, it will recommend the most efficient option available. This will ease the load on DevOps in their capacity planning for their cloud and other such resources, and provide significant cost-savings for the application team. In this manner, AIOps will be useful for an organization to reduce their software operational costs.

3.8. Release Management

When new software or firmware is deployed, it is often done in a phased manner. After a small portion of deployment occurs, AIOps can be used to analyze the difference between the new deployment and the previous version using machine learning models. If there are some significant changes, the model will identify which parameter or characteristic has changed. These insights would help the operations team to make a decision on whether to continue with the deployment or to roll back to a previous version.

4. Anomaly Detection

Anomaly detection is the key for proactive monitoring. It can be used for data quality monitoring, and fault detection of IT operations. Anomalies can be observed in tabular data, timeseries, or temporal data, as well as in graphical images. However, when it comes to fault detection in IT operations, data is typically in the form of time-series and so time-series anomaly detection is performed in AIOps.

Anomaly detection refers to identifying any abnormal behavior or pattern of data from the learned normal pattern from historical data. These anomalies may indicate a problem or an interesting event. The learned model can be used to detect anomalies with varying degrees of probability, and to predict future data with certain confidence.

4.1. Anomaly Detection Algorithms

When it comes to detecting anomalies in time-series data, the task can be performed either in a supervised, a semi-supervised, or an unsupervised fashion depending upon the dataset.

(1) Supervised anomaly detection is possible when we have annotated data of anomalies available. We can split the data with anomalies into train and test data, and train a machine learning model as a binary classification problem, which means, training a model to predict whether a point is an anomaly or not using the labels available as feedback to train the model. This method can be performed using any machine learning model that can be used for binary classification such as Deep Neural Networks (DNN), Support Vector Machines (SVM), Random Forest, Gradient Boosted Tree (GBT), eXtreme Gradient Boosting (XGBoost), and others.

(2) Semi-supervised anomaly detection can be performed when we do not have labelled anomalies, but we have a significant amount of data that is normal and do not have many anomalies that we can use for training a model. This model is trained supervised using the normal data and it learns the normal data behavior. It then detects anomalies when any deviations are observed. A DNN model like Autoencoder, which is a kind of neural network that learns a pattern and tries to replicate it, can be used for this. We can also use other DNN models like Long short-term memory (LSTM) and Deep convolutional neural network (CNN) (e.g., DeepAnT [3]). One-class SVM, Gaussian Mixture Model (GMM), Kernel Density Estimation (KDE), and others can also be used for performing semi-supervised anomaly detection.

(3) When we do not have labelled data, as in most practical situations, we use unsupervised anomaly detection techniques. In this, the train or test data may or may not have anomalies. The models in this class generate an anomaly score for every data point and we can select a suitable threshold depending upon the data to classify points as anomalies or not. This can be accomplished using statistical methods like simple Moving Average model or complex ones like Prophet Forecasting Model [4] (uses Fourier series) or SARIMAX (Seasonal Auto-Regressive Integrated Moving Average with eXogenous factors, an extension of the ARIMA model - Auto-Regressive Integrated Moving Average). These approaches learn the periodicity or seasonality and the trend of the time-series data, and detect when the data deviates from the normal pattern. Density or distance-based outlier detection algorithms like k-Nearest Neighbors (kNN), Isolation Forest, or Local Outlier Factor can also be used to perform unsupervised anomaly detection.

With users' feedback, the detected anomaly can be labelled as true or false detection, and stored in the database. The historical annotations of ground-truth labels can be useful to evaluate the performance of the anomaly detection model in terms of evaluation metrics like Precision, Recall, and F1-Scores [7]. Also, they can be used to train a classification machine learning model for performing supervised anomaly detection. By getting feedback from the user (an

operator) for false positive detections, we can also adjust the threshold for anomaly detection, which is particularly beneficial to improve the accuracy for semi-supervised and unsupervised models.

4.2. Anomaly Detection Platform

We built an anomaly detection platform for proactive monitoring (Figure 2) with the following functionalities.

(1) ***Algorithm selection:*** We allow the user to select among multiple algorithms, which run in real time by learning the seasonality and trend from data. We have options to perform either unsupervised, semi-supervised, or supervised anomaly detection depending on the use case.
(2) ***Special events:*** There can be cases when irregularity is expected on days like deployments, holidays such as Christmas, or special events such as a key NFL game. Anomaly detection algorithms, like Prophet or SARIMAX, will consider these as special events. We have the option in our AIOps platform to provide such dates in advance to prevent false alarms as the model treats such days differently than a normal day.
(3) ***User feedback:*** The tool can also take feedback from operator if they feel that certain prediction points are false positives, i.e., false alarms or shouldn't be anomalies from their subject knowledge.
(4) ***Intelligent alerting:*** We provide three rules for the user to configure alerts. They are:
    a. ***Everytime*** – To trigger alert notification everytime a model detects an anomaly.
    b. ***Interval Threshold*** – To send alert notification only if the percentage of anomalies detected by a model out of all the data points present over a certain time period (10 minutes, 1 hours, daily) specified by a user, exceeds a certain threshold (0% – 100%) which is specified by the user as well.
    c. ***Score Threshold*** – To send alert notification if a model predicts an anomaly score (based on model confidence) greater than a specified score threshold which is greater than or equal to the anomaly detection score threshold.

    We allow the user to configure one or more such alerts for a single prediction model depending upon their needs. We also let the user choose the severity of the alert (low / medium / high) which they can configure accordingly.
(5) ***Messenger notification:*** We send alerts to the operations team via messenger with a snapshot of the data (and potentially root cause analysis report) and a link to the data dashboard. We also provide 'Pause Alert' buttons at different durations for the operator to pause alerting for a specific metric prediction.
(6) ***Alert history:*** We store all the historical alerts for all metric predictions for the user to view from the dashboard anytime. In addition, we also allow the operations team to collect the alerts as a metric data into their metrics endpoint, if required.

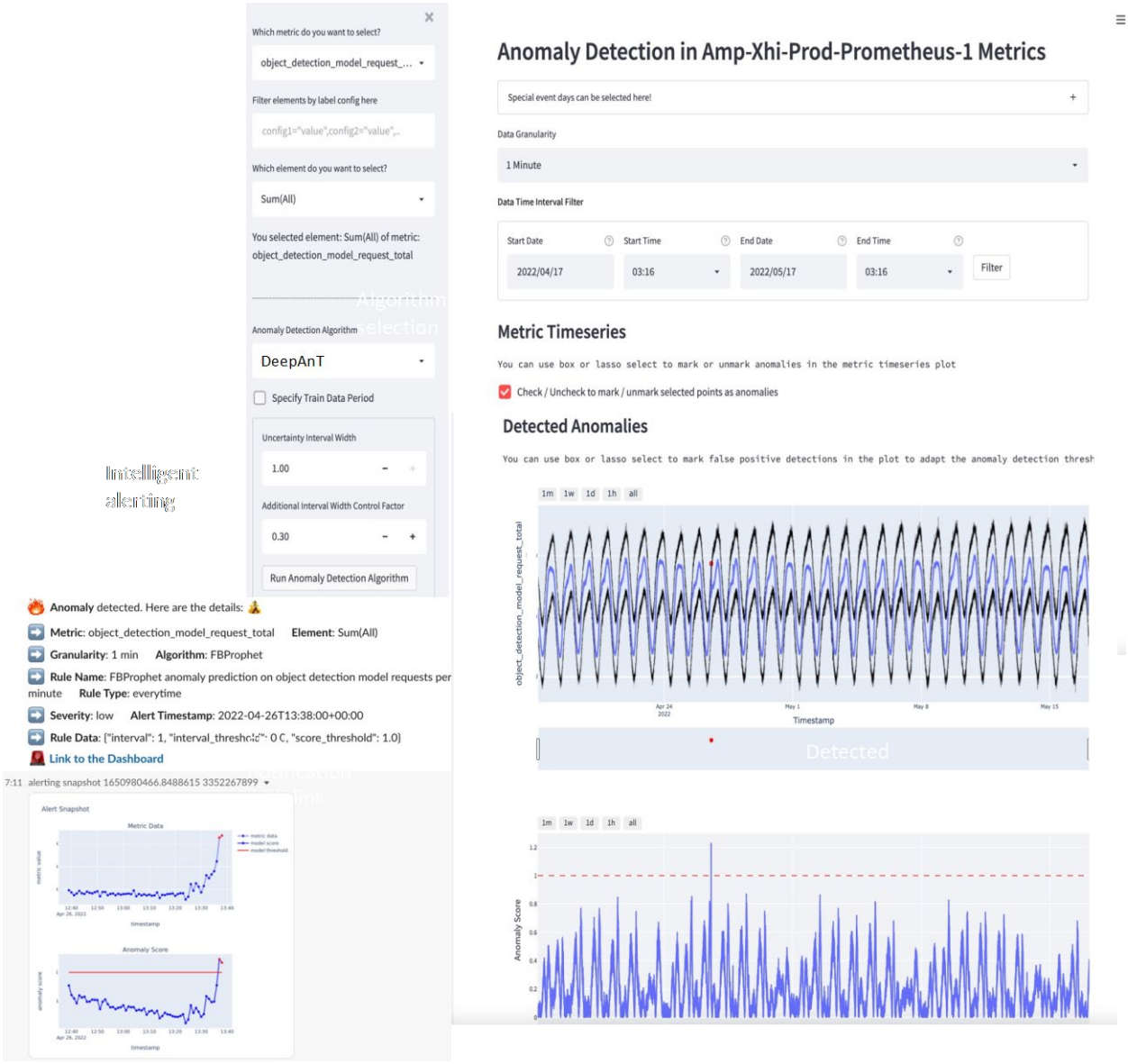

Figure 2 – Anomaly Detection Platform

5. AIOps – Operators' Perspective

This section describes AIOps from an operator's perspective as they will be the primary users of our platform, interacting with it in their routine work. This includes things that they need to know to onboard their application onto the AIOps platform, to configure anomaly thresholds, to provide feedback while evaluating a machine learning model against a specific metric, and to configure alerts for notification.

5.1. Onboarding a tenant

To onboard an application or a tenant, the AIOps tool needs read-only access to metrics data and log data sources. Log data is required if we are enabling automatic log mining to correlate with anomaly metrics. Operators can view the list of available metrics from an end point and start experimenting with specific metrics. After enabling the permission for AIOps to fetch data from their application tools, they need to provide a config file stating which metrics to pull and the endpoint configuration related details. Upon submission of this, the AIOps tool starts the data ingestion in real time.

5.2. Training a metric with model

The next step for an operator after onboarding an application is to configure an AI model. For that, they can choose the metric of interest and preview the metric trend over a selected time period. They can then preprocess the data, like aggregations, over a period and experiment with the choice of machine learning algorithms available in the AIOps platform.
The operator can perform model training live on the chosen data and preview the results on the fly. They can provide feedback like false positives or negatives to fine tune the model. After this, they can also validate the model with some test data on a certain duration of data. This process can continue until the operator is confident with the results for the data selected. Finally, if satisfied, the model can be packaged and deployed as it is ready for real-time inference.

5.3. Schedule a metric for real-time inference and prediction

With the AIOps platform, the operator can schedule one metric or a group of metrics for real-time inference configuring the frequency in any duration. Once configured, the AIOps platform will continue to run the selected model for metric anomaly analysis in real time.

5.4. Configure Alerts and RCA

Alerts can also be configured for the detected anomalies along with the severity and the considered interval or a model score threshold. The corresponding alerts can be notified to clients using messenger via their web APIs.
Root cause analysis can be performed automatically where the system performs correlation calculations on multiple anomalies detected on the metrics around pre-selected specific periods. This is performed by correlating the anomalies detected with the errors observed in log data in order to identify the most probable source error logs that could have caused the anomalies through a scoring mechanism which helps to rank the root cause errors. The AIOps tool recommends a possible hypothesis that corresponds to the most probable root cause of the problem which an operator can easily pick up for subsequent actions towards resolution. In this way, MTTR can be reduced greatly by using our AIOps platform.

6. AIOps Case Study: Connected Living Object Detection Operation

For our Connected Living business, we have millions of cameras in customers' homes. Our customers would like to get notification when objects (e.g., person, vehicle, and pet) or events (e.g., package delivery) of interest are detected from their cameras. The AI for Connected Living

team built the state-of-the-art house object detection algorithm which is used to efficiently detect person, vehicle, and pet.
The main challenge for the object detection operation are as follows.

(1) There are many metrics to keep track of such as load (request per second), latency (upstream, downstream, and inference), CPU, and memory for each node.
(2) The load (request per second) changes dramatically between day and night. There is a greater load in the daytime than in the night which is reasonable as generally human life is busier in daytime. The previously used static threshold rule-based alerting is not adaptive to address this issue.
(3) The application has lots of log data. It logs the interactions of the object-detection module with the input metadata (from camera and the backend platform). Once there is an alert, the operations team often needs to dig into this log data to identify the root cause.

We onboarded this application onto the AIOps platform to help the operations team by addressing the above challenges. In that, we deployed time-series anomaly detection algorithm to alert the operations team in messenger, and then correlated the detected anomaly with log metrics anomaly to report what error messages are the probable root cause.

6.1. Connected Living Object Detection Operation

The metric data[1] of concern for this case study is the number of object detection model requests per minute. This metric captures the information of the load and has a well-defined daily seasonality pattern as we can see from the sample metric time-series plot shown below in Figure 3.

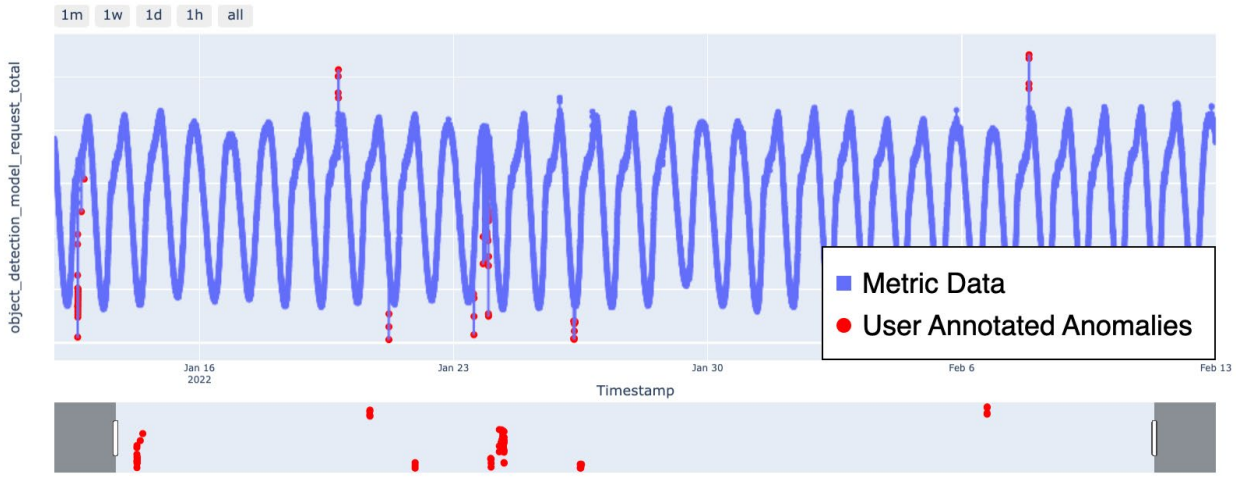

Figure 3 – Metric in Object Detection Operation

In the Figure 3, we can observe the metric time-series data plotted in blue. As we can observe, there are situations that lead to some sudden dips or spikes in the metric values that are of concern for the operations team to monitor. The points marked in red dots are the annotations

[1] We collect, store, and use all data in accordance with our privacy disclosures to users and applicable laws.

provided by a user by using a lasso or box selection tool available in the plot in the dashboard. By default, the metric data fetched from the metrics data endpoint will be unlabeled.
Since this metric data is unlabeled and it has a well-defined seasonality, we chose an unsupervised anomaly detection algorithm for this case study. Prophet [4] is one algorithm that can be used for univariate time-series forecasting as it fits on the historical data and forms an additive model of the trend, daily, weekly, and yearly seasonality components, as well as holiday effects, and additional regressors that are available and learned from the data. By fitting such a model over this data for a training period of at least two weeks, we get an accurate model that learns the seasonality and forecasts with anomaly scores that can be derived from the uncertainty bounds generated by the model. The anomaly score is controlled by the interval width factor (which represents the percentile values) and an additional multiplicative factor controlling the width of the model prediction bounds. Since the model generates samples by Maximum A Posteriori (MAP) or Markov Chain Monte Carlo (MCMC) sampling techniques, along with the expected predicted value, we can also get the percentile values that are used for generating the uncertainty bounds and in turn, the anomaly scores. We can observe the predictions made by the model in the following time-series plot in Figure 4.

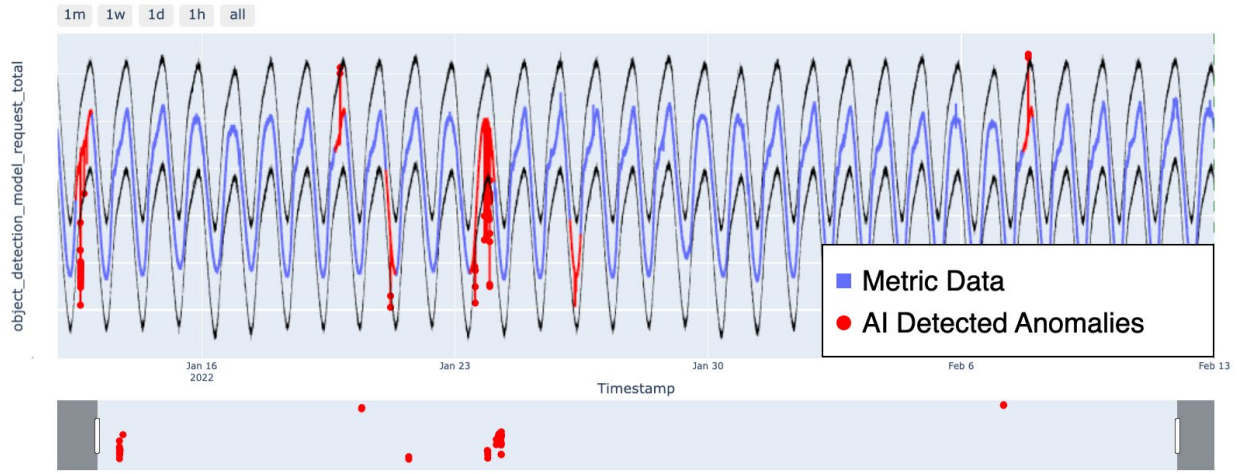

Figure 4 – Anomaly Detection Predictions in Object Detection Operation

We can see the points that are marked in red, and they represent the anomaly detections made by the model. Since the ground-truth labels are annotated in the metric plot, the sections of the time-series data where labels are provided have been marked with red lines while the other sections are in blue. These sections represent a tolerance interval to evaluate the model performance in terms of Precision, Recall, and F1 Scores. Having a tolerance is generally followed for time-series anomaly detection evaluation as forecasting anomaly in advance proactively, or after a pre-defined delay, is generally acceptable [5]. So, predictions happening within such contiguous tolerance intervals are all considered as precise predictions while evaluating a model's performance. Since this is followed in literature and by popular service providers, we have followed a similar approach.
The model evaluation is subjective to the dataset and what the operator choses to mark as actual anomalies. Therefore, we cannot have evaluation metrics for all the models at all data periods. In

this section of the data shown in Figure 3 and Figure 4, however, the evaluation metrics are as follows:

- Precision: 1.0
- Recall: 0.87
- F1 Score: 0.93

Just as we can annotate labels in the metric plot shown in Figure 3, we can also annotate false positive predictions in the plot shown in Figure 4. That feedback is used to adjust the anomaly score thresholds to predict those points as normal. We fitted our model on this dataset and adjusted the thresholds to have a good prediction model that can accurately detect anomalies and alert the operations team for further investigation by setting an alert after the training and fine-tuning activities of the model are completed. The alerts are sent to a messenger channel with the information of the alert and the metric data and provides a snapshot of the metric and anomaly scores. There is also a shortcut link provided to the operator for them to look at the data dashboard directly from the message. As we can see from a sample snapshot of the messenger alert shown in Figure 2, the operations team was notified of the alerts in real time by the application and they were able to check why the object detection requests suddenly shot up. This real-time data pulling, model inference, and smart alerting capabilities that are provided by the application helped to ease the monitoring task performed by the operators for this case and provided timely alerts for them simplifying their operations.

6.2. Log mining and Correlation Analysis

Beyond anomaly detection and timely smart alerting, one of the common situations faced by an operations team when they encounter such alerts is finding the root cause for an anomaly. This operation is not straightforward, and the DevOps team will have to manually look over the log messages to identify the errors and warning messages in the logs within the period closer to the timestamp when the alert occurred. This manual operation is usually time-consuming and may also critically impact the businesses if the resolution or remediation cannot be taken within a certain time. A sample of such an operation has been shown in Figure 5.

Figure 5 – Manual Log RCA in Object Detection Operation

In this Figure 5, we can see that an alert has been sent to the operations team of an anomaly detected in the object detection requests per minute metric. In order to find the root cause, the log messages in the dashboard were scrutinized and the team was able to identify a unique error log message that was logged near the timestamp when the anomaly was detected. As we can see that the requests had suddenly dipped, we observed that it was due to a deployment that had taken place at that time that was causing some errors from the Kubernetes pods running the object detection model containers. To reduce this manual effort, for this case study, we worked with the operations team to help them solve this problem using intelligent log mining and automatic root cause error analysis, in addition to anomaly detection and alerting.

In our application, we added the feature to pull and store log messages in addition to pulling and storing metrics data. We do not store every log message but only error and warning log messages through smart keyword searches for such terms in the logs. Apart from pulling and storing the logs, we also do intelligent text parsing to identify different kinds of log messages. With this method of segregating the error logs and labeling each type uniquely, we create error log count metrics for each error log message type, and store them as a separate metric.

So, the outcome of intelligent log mining provides us the error log metrics that can be used to individually train anomaly detection models to predict and alert whenever a pattern changes or an appearance of an error log message occurs. In addition to performing anomaly detection on them, these metrics can also be used for root cause analysis when anomalies are detected in metrics as we have a mechanism to correlate anomalies detected on a metric with other metrics over a selected data period and provide a ranked list of correlated metrics as probable root causes, as we'll see in the next case study.

Even though the error log metric creation gives us two potential use cases, the primary objective of log mining is to perform intelligent instant root cause analysis directly from the log messages when anomalies are detected in metrics, and to notify the operator of the alert along with the probable root cause error log messages in the messenger notification. For this, we have some scoring mechanism that is used to rank the error log messages that appear in the recent past period from the time the alert was identified, by comparing the frequency and distribution of the same over a much larger previous historical period. This helps us provide the operations team with the limited set of the most probable root cause errors instantly that they can quickly identify and perform remediation. The dashboard has the option to let the user select any alert that occurred in the past and view a more detailed root cause analysis report by showing the error log trends of each of the recently observed error logs as well the raw log messages as shown in Figure 6. This implementation reduced the time taken for root cause analysis tremendously and helped the operations team to be more productive.

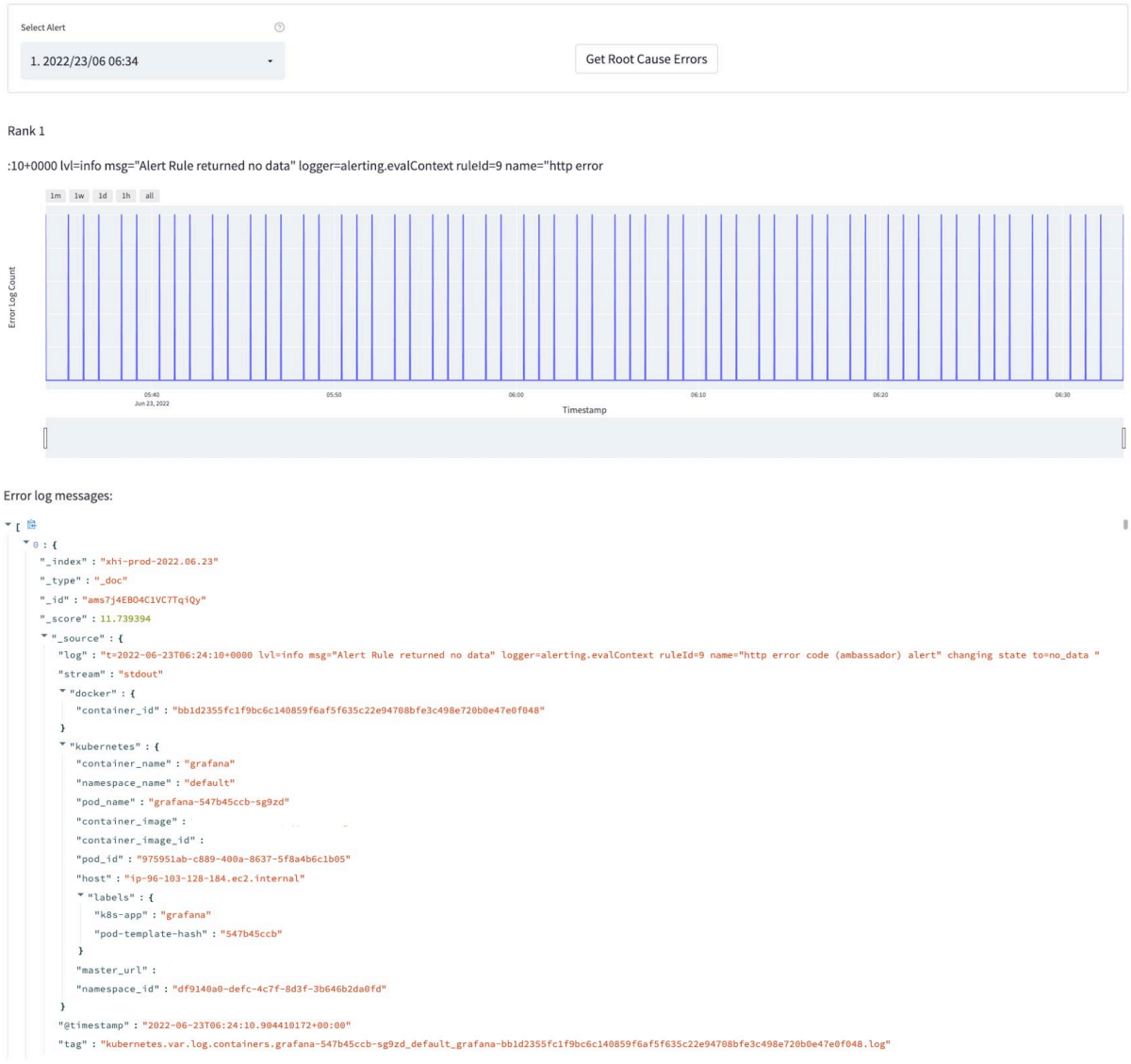

Figure 6 – Automatic Log RCA in Object Detection Operation

7. AIOps Case Study: Rex Browse and Search Operation

We have a large number of customers using an Xfinity box for cable TV and video streaming purposes. Here, the Rex platform is providing the video discovery features for X1 boxes. Specifically, Rex offers Keyword Search and Menu Browse services to X1 video clients. The platform also provides support for Video Recommendations, Personalization services and Video Metadata service.

Rex is deployed in several datacenters across the globe to cater to our business needs. Due to this, the DevOps team can handle significant complexity between various system metrics flowing from several sources. Their major pain point is performing RCA, especially in critical situations when there's time-constraint if the issue has a direct customer impact.

Let's take a look at an example using the below Figure 7.

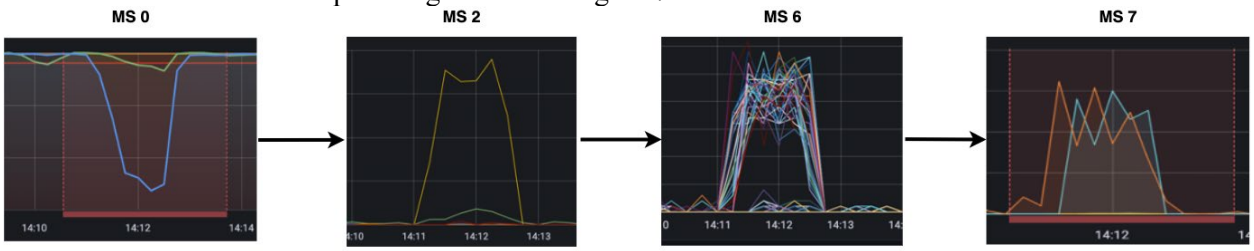

Figure 7 – Manual Flow of Issue Root Cause Diagnosis by Rex Operations Team

Figure 7 shows the investigative steps performed by the operations team after a long diagnosis of an issue (dip in a system metric) affecting customers using Rex features. We can observe one of the regular flows of issue triaging and diagnosis performed by the Rex operations team in case of an issue. At each of those steps, the operator has to go through hundreds of metric graphs before drilling down to the next level of the problem. This process involves significant delays and inefficiencies. Because it is a customer facing application, time is of the utmost importance and MTTR converts to business value; the lower the MTTR, the better the customer experience. Through the AIOps tool, we provided features which greatly assisted the operations team in their routine work as we'll see in next sections.

7.1. Dependency Graph

The dependency graph of a system denotes the hierarchy in which different services inside the system are interlinked with each other. The network calls go from the top layered service to bottom layers according to the service dependencies. Let's look at a structure of a graph we had generated in a Rex case study. We have masked the names of the actual micro services used in Rex and have provided the contextual details alone for confidentiality concerns.

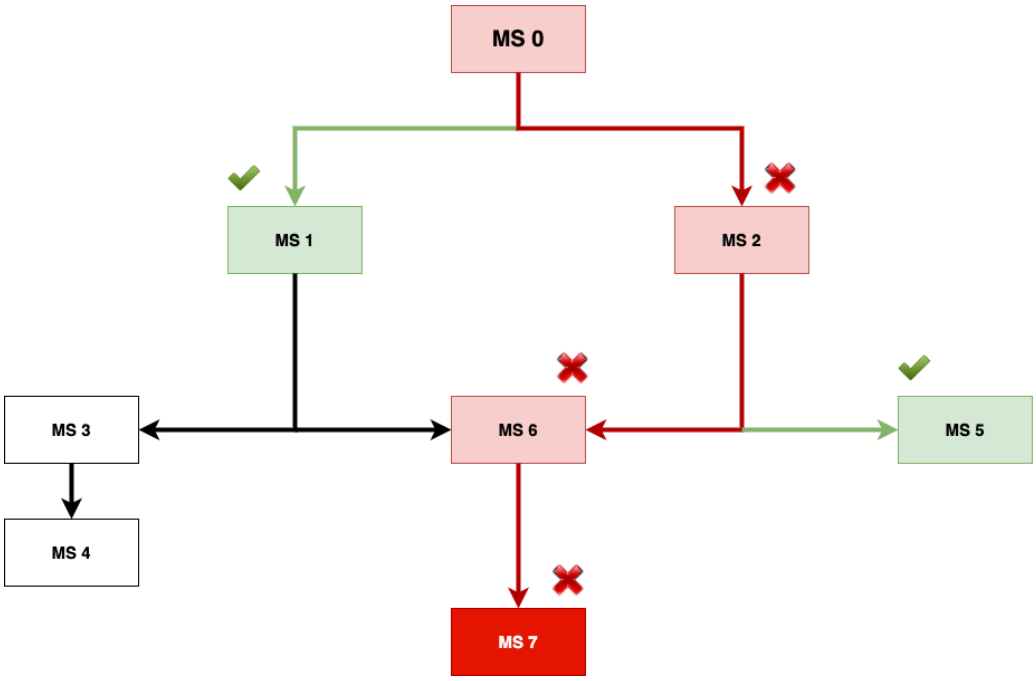

Figure 8 – Dependency Graph Created in Rex Case Study

In Figure 8, we can see the dependencies among different services (MS – Micro Service, used in the general sense of software engineering literature) and also the error flow denoted by color code (red – error, green – ok). This graph was constructed by our AIOps team using domain expertise from Rex technical operations team. We aim to automate the process of generation of such graphs in upcoming releases of the AIOps platform.
The benefits of having such dependency graphs for operations work are:

- This can be used by operations team to have a bird's-eye view on the entire system.
- Using such a hierarchical system topology i.e., hierarchy within system services, and applying RCA at each level, will help in swift identification of root cause. Also, the error flow captured will help the operator perform faster and detailed resolution.
- In this way, we are reducing the time and effort required by DevOps in their normal work, thereby reducing MTTR.
- Also, this graph can be used for efficient scheduling of on-call rotations since we know from the diagram what the affected systems are, and can estimate the efforts and expertise required to address those cases.

In the next section, we will see how we leverage AI techniques in our platform to help the Rex team with RCA.

7.2. Root Cause Analysis with Dependency Graph

As we observed in the earlier sections, manual RCA work required a great deal of time and effort by the operations team, increasing the time taken for tracing and resolving the issue. We have a feature for automating the RCA task in our AIOps platform. We performed anomaly detection on all the relevant metrics that we ingested into the platform. These are the metrics used for monitoring by the Rex operations team and are mostly custom configured metrics using queries. We then correlated the anomalies detected in one metric over a time period with those of other

metrics and generated a ranked list of probable root causes. Figure 9 shows the results of performing automatic root cause analysis using the AIOps tool for Rex case:

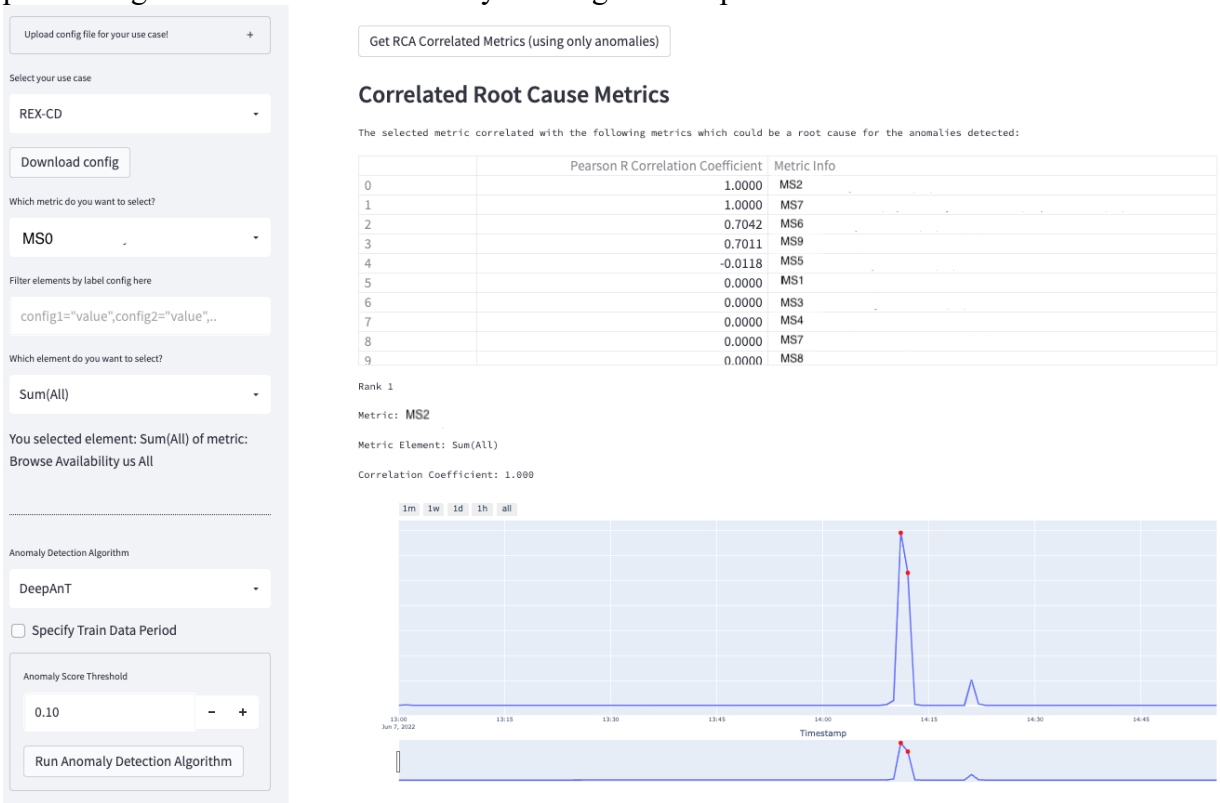

Figure 9 – AIOps Showing Automatic RCA Correlated Ranked List

As we can see from Figure 9, the tool showed the ranked list of metrics whose detected anomalies correlated with the anomalies detected on the selected metric. If we take a metric from one microservice, say MS0, when anomalies are detected and alerts are notified, the operations team can check for all other metrics from other microservices like MS1, MS6, and others where we observe correlated anomalies.

The AIOps tool shows the Pearson correlation coefficient score in the ranked list for the operator to understand how well the metrics correlate. In addition to showing the ranked list, the tool also shows the individual metric time-series plots of the correlated metrics in the same ranked order, for the operations team to quickly verify the correlations and drill down further. We can see the further list of correlated metrics in Figure 10.

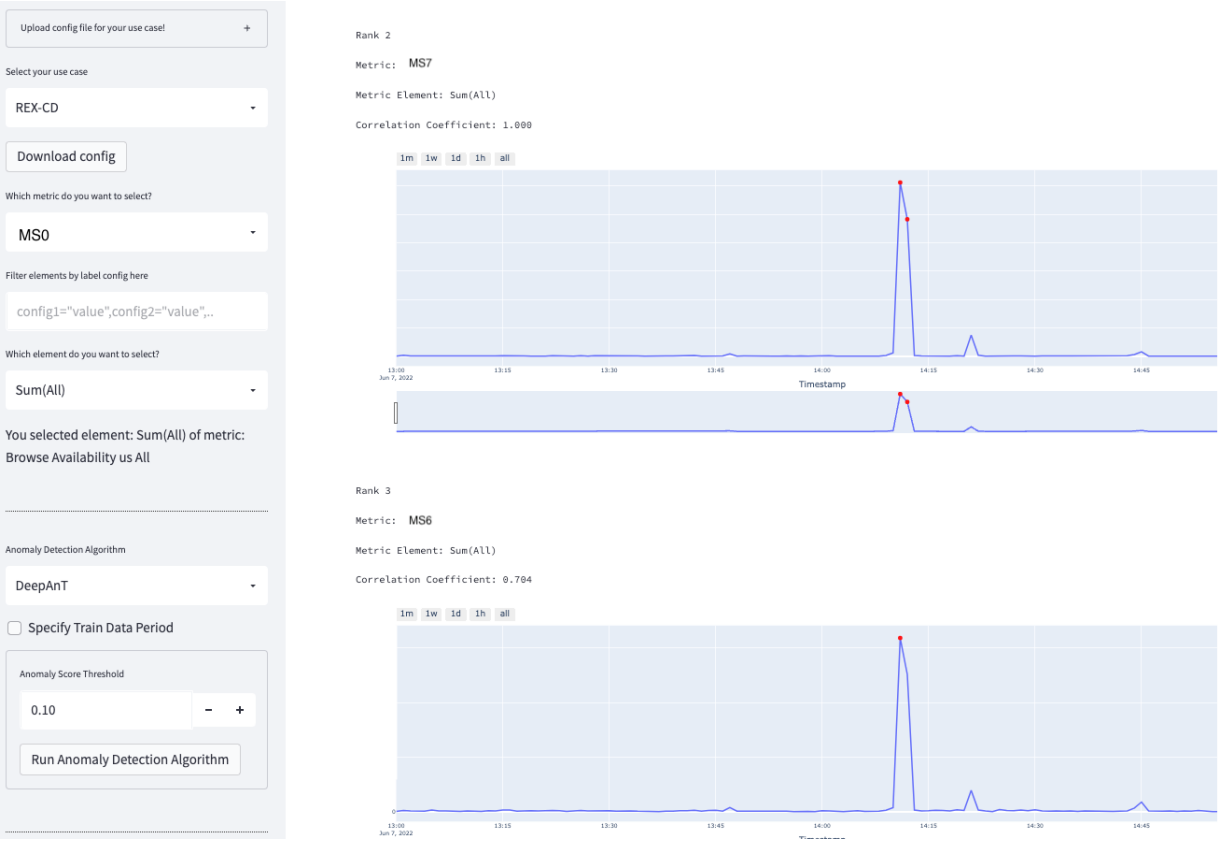

Figure 10 – AIOps Showing Probable Root Cause Metric Time-series

As per the manually identified dependency graph, we observed the root cause metrics to correlate well with the parent metric. As automatic dependency graph generation from user input is something we have in our pipeline, currently the correlations are shown across multiple levels of the hierarchical topology. However, we plan to improve the application to perform this root cause analysis at each level with only their dependent metrics (immediate child nodes) and after identifying the most probable root cause(s) with some threshold, automatically drill down further from that level and so on using dependency graph traversal as shown in Figure 8, instead of performing a correlation over all the metrics across all levels of hierarchy. This way, the operation will be optimized in the application when we have a smaller beam width for searching.

7.3. Failure Prevention and Auto Remediation

The Kubernetes pods run on Java Virtual Machine (JVM). They periodically collect garbage when requests sent by clients back up and physical memory runs low. Excessive Garbage Collection (GC) can also be a sign of the overall service going into a bad state.
One or two pods collecting garbage at one time is not problematic, however GC running for hours unchecked in a self-reinforcing loop can infect other nodes. It is at this stage that drops in a metric like availability become noticeable. Any pod doing abnormal GC should be restarted after a preset period, but only if the outage is isolated to it.
To accomplish this goal, we maintained a count of nodes doing GC at any given time. A two-pronged approach was used to highlight pods that can be restarted:

- First, we ruled out a system level outage by looking at the percent of pods doing GC as shown in Figure 11 . The normal threshold can vary by the microservice and data center. Some data centers which are physical and running older, slower machines can be more failure-prone than others. Some microservices may be more memory intensive. Checking every such combination manually was not possible.

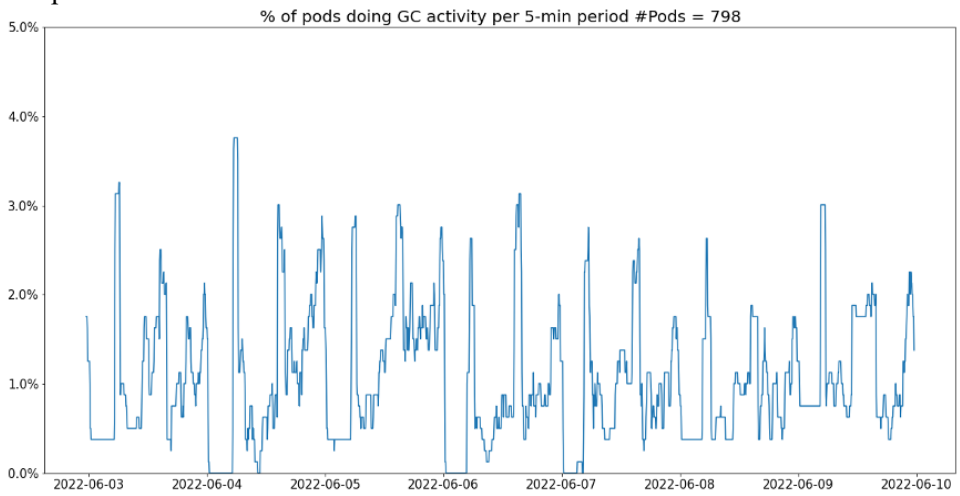

Figure 11 – Percentage of Pods Having GC activity per 5-min Period

- Second, assuming that there was no system level outage, we sought to find the pods with anomalous GC activity. This is a user input, defaulted to x standard deviation multiples of normal GC activity. The on-call messenger channel was alerted with the IP addresses of pods exceeding this threshold with the suggestion to reboot. The kill signal was also sent automatically as part of an auto remediation AIOps use case.

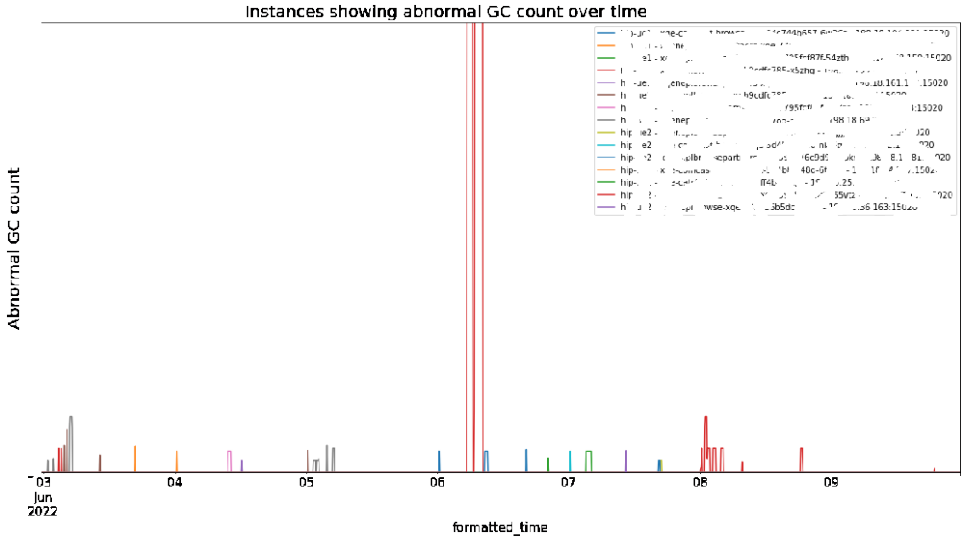

Figure 12 shows some of the malfunctioning pods. We had significantly narrowed down the list of potentially malfunctioning pods to around 2% of all pods, saving precious time during monitoring.

8. AIOps Case Study: Release Management for RDK Firmware

When a new RDK firmware version is released for Camera (RDK-C), Broadband (RDK-B) or video (RDK-V), we usually adopt a phased rollout. Usually, a small portion of customers with random selection (1%, or 5%) will be first deployed with the new firmware, and the operations team will closely monitor potential performance issues.
The challenges lie in several aspects:

(1) There are often increasing device-specific issues, such as performance or stability, and at the same time there are hundreds of telemetries and parameters for the operations team to track. Some key parameters to monitor for each firmware version include VOD (Video on Demand), Linear, WHIX (Wi-Fi Happiness Index) [8] , SpeedTest, CPU (Central Processing Unit), and Load among others.
(2) Identification of new release issues are heavily dependent on Call-In-Rate (CIR) and high-level call movers/truck rolls, i.e., no-signal and no-block-sync. This results in long lead time for another iteration.
(3) RCA and triaging need manual review of multiple boxes/examples. This is time consuming.

The RDK team deployed our release management AI component to drive release decisions in an automated manner. We can evaluate the impact on sub-populations (e.g., targeted/control for the specified segment such as region, CMTS (Cable Modem Termination System) version, HDCP (High-bandwidth Digital Content Protection) version, or accounts with pods) and firmware segments. Thus, the operations team can identify the anomalies with limited deployment, identify the root cause, and respond to the events quickly.

8.1. Machine Learning Model

We built a non-linear classification model to classify gateways with two firmware versions using one day's data where a new firmware version was deployed into the field. We focused on gateways with the same model to minimize hardware specific difference, and we additionally balanced the dataset such that the quantity of gateways with old version was roughly the same as those with new version.
The labels to the classifiers were the firmware versions. There are two major types of model features: counts of RDK-B telemetry key occurrence and certain measures on gateway usages (e.g., CPU usage, memory usage, Wi-Fi signal strength). Features were engineered from telemetry aggregated within a time window of 24 hours. Specifically, for a gateway with new version, features were collected within the 24 hours when the new version was deployed, and during the same 24-hour period, features were also collected for gateways with the old version. Our hypothesis was that if the version changes caused unexpected errors or performance change, then the classifier using RDK telemetry would be able to accurately differentiate the two firmware versions and indicate what telemetries were significantly impacted by the version change, and possibly point to the root cause.

8.2. Model Results

We built the model on a sample of approximately 200K gateways of a specific type. Our model was able to differentiate the two firmware versions with a high accuracy (Figure 13 Left). Based on feature importance score from the model, we were able to rank the features based on their

impact on the model performance. One of the top ranked features was the Wi-Fi signal strength on 5G band (5GRSSI_split). We further investigated that feature and found that there was a significant shift between overall signal strength distribution before and after the version change (Figure 13 Right). Here the old firmware version is 3.3p19s1, and the new one is 3.4p3s1.

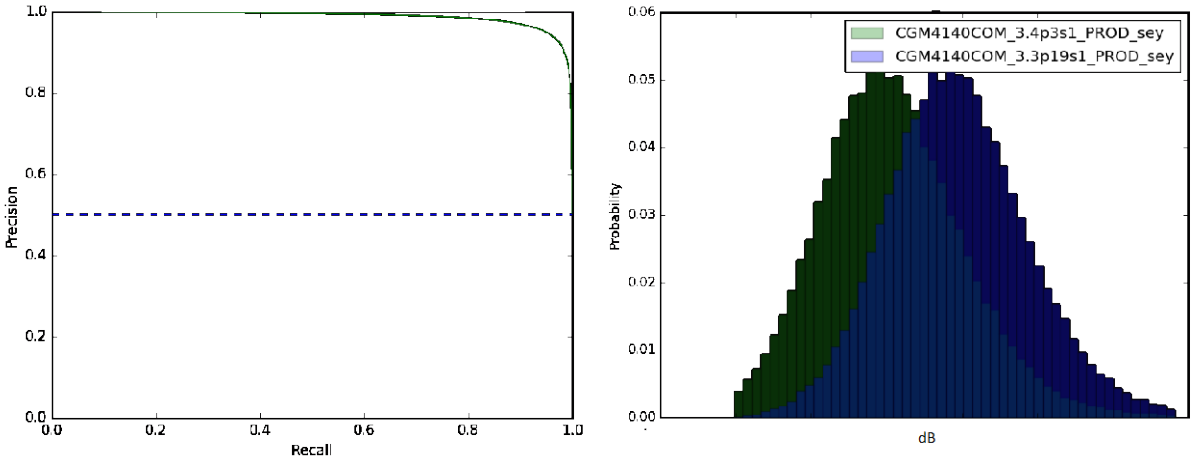

Figure 13 – Precision Recall Curve for the model (Left Figure); 5G WiFi Signal Distribution for Old Firmware Version and New Version (Right Figure)

We confirmed the change with RDK-B team and learned that the shift in distribution was due to a bug fix on the reporting of Wi-Fi signal strength, i.e., old version tended to artificially report better signals with a 10 dB difference. The machine learning model we developed here not only helped us to gauge whether a new firmware release might cause significant difference in gateway performance, but also helped RCA for further investigations.

9. Conclusions

We presented an introduction to AIOps, and discussed our AIOps platform, which includes anomaly detection, root cause analysis, and release management, among many other use cases. We onboarded three applications using the AIOps platform and demonstrated the effectiveness of the platform regarding improvement in the experience and productivity of operation teams, and potential reduction in operational cost. As next steps, we would like to further evaluate the impact of AIOps quantitively.

10. Acknowledgements

We would like to thank the following teams for the discussion and close collaboration with AIOps team to onboard the application: (1) AI for Connective Living team, especially Noam Krendel, Don Tolley, and Luke DeLuccia. (2) Rex team, especially Li Cao, Dirk Walter, Luciano Polo, and Albert Ogonevskiy for brainstorming the use case, and (3) RDK Analytics team, especially Matthew Long. We also thank Jan Neumann and Amit Bagga for their guidance.

Abbreviations

5G	5th Generation
AD	Anomaly Detection
API	Application Programming Interface
ARIMA	Auto-Regressive Integrated Moving Average
AI	Artificial Intelligence
AIOps	Artificial Intelligence for Information Technology Operations
CAGR	Compound Annual Growth Rate
CD	Continuous Delivery
CI	Continuous Integration
CIR	Call-In-Rate
CMTS	Cable Modem Termination System
CNN	Convolutional Neural Network
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
DeepAnT	Deep Learning-based Anomaly Detection for Time-series
DevOps	Software Development and Information Technology Operations
DNN	Deep Neural Network
e-mail	Electronic Mail
GMM	Gaussian Mixture Model
GA	General Availability
GBT	Gradient Boosted Tree
GC	Garbage Collection
HDCP	High-bandwidth Digital Content Protection
IIM	Intelligent Infrastructure Monitoring
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
JVM	Java Virtual Machine
KDE	Kernel Density Estimation
kNN	k-Nearest Neighbors
KPI	Key Performance Indicators
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
MLOps	Machine Learning Operations
MS	Micro Service
MTBF	Mean Time Before Failures
MTTR	Mean Time To Resolution
NFL	National Football League
NLP	Natural Language Processing
RCA	Root Cause Analysis
RDK	Reference Design Kit
RDK-B	Reference Design Kit for Broadband
RDK-C	Reference Design Kit for Camera

RDK-V	Reference Design Kit for Voice
RM	Release Management
RSSI	Received Signal Strength Indicator
SARIMAX	Seasonal Auto-Regressive Integrated Moving Average with eXogeneus factors
SCTE	Society of Cable Telecommunications Engineers
SMS	Short Messaging Service
SVM	Support Vector Machine
TSDB	Time Series Database
VOD	Video On Demand
WHIX	Wi-Fi Happiness Index
Wi-Fi	Wireless Fidelity
XGBoost	eXtreme Gradient Boosting

Bibliography & References

[1] How Much Time Are You Wasting on Manual, Repetitive Tasks? Survey by Smartsheet https://www.smartsheet.com/content-center/product-news/automation/workers-waste-quarter-work-week-manual-repetitive-tasks
[2] AIOps Platform Market Forecast to 2028 - COVID-19 Impact and Global Analysis by Component, Deployment, Organization Size, and Vertical, Reportlinker, Apr. 2022.
[3] M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," in IEEE Access, vol. 7, pp. 1991-2005, 2019, doi: 10.1109/ACCESS.2018.2886457.
[4] Taylor SJ, Letham B. 2017. Forecasting at scale. PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2
[5] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at Microsoft. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3009–3017, 2019.
[6] Pearson correlation coefficient wiki page, https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[7] Precision, recall and F1-score wiki page, https://en.wikipedia.org/wiki/Precision_and_recall
[8] Krithika Raman, Charles Moreman, THE WiFi Happiness Index (WHIX), SCTE Fall Technical Forum, 2011

