


Incident Management for Explainable and Automated Root Cause Analysis in Cloud Data Centers


Arnak Poghosyan

(Institute of Mathematics of NAS RA, Yerevan, Armenia
VMware, Palo Alto, US

 <https://orcid.org/0000-0002-6037-4851>, arnak@instmath.sci.am, apoghosyan@vmware.com)

Ashot Harutyunyan

(Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia
VMware, Palo Alto, US

 <https://orcid.org/0000-0003-2707-1039>, aharutyunyan@vmware.com)

Naira Grigoryan

(VMware, Palo Alto, US
ngrigoryan@vmware.com)

Nicholas Kushmerick

(VMware, Palo Alto, US
nicholask@vmware.com)

Abstract: Effective root cause analysis (RCA) of performance issues in modern cloud environments remains a hard problem. Traditional RCA tracks complex issues by their signatures known as problem incidents. Common approaches to incident discovery rely mainly on expertise of users who define environment-specific set of alerts and target detection of problems through their occurrence in the monitoring system. Adequately modeling of all possible problem patterns for nowadays extremely sophisticated data center applications is a very complex task. It may result in alert/event storms including large numbers of non-indicative precautions. Thus, the crucial task for the incident-based RCA is reduction of redundant recommendations by prioritizing those events subject to importance/impact criteria or by deriving their meaningful groupings into separable situations. In this paper, we consider automation of incident discovery based on rule induction algorithms that retrieve conditions directly from monitoring datasets without consuming the system events. Rule-learning algorithms are very flexible and powerful for many regression and classification problems, with high-level explainability. Since annotated or labeled data sets are mostly unavailable in this area of technology, we discuss data self-labelling principles which allow transforming originally unsupervised learning tasks into classification problems with further application of rule induction methods to incident detection.

Keywords: data center management, performance incidents, anomaly detection, root cause analysis, machine learning, rule induction, RIPPER, JRip

Categories: I.2.6, I.5.3, I.5.4

DOI: 10.3897/jucs.76608

1 Introduction

Automated and intelligent root cause analysis (RCA) is of great importance for proactive detection and self-remediation of IT issues in large-scale distributed cloud environments before they impact an end-user via application/infrastructure performance degradation. One of the common approaches to RCA is a problem detection and identification by its accompanying group of symptoms known as fingerprint or incident. Automated discovery of data center symptom-patterns is recently gaining a lot of popularity. A problem incident is a group of symptoms (alerts, alarms, events, metric changes, property changes, etc.) with sufficient historical evidence in reoccurrence and similarity. All discovered incidents should be stored in a knowledge-base together with available expert annotations regarding the problem description and its possible resolutions. The knowledge-base of incidents with sufficient coverage of system's possible failures is an invaluable asset for any system administrator and one of the important components of data-analytics for a self-driving data center.

The most challenging vision of IT management is realization of AI-powered self-driving data centers with predictive maintenance, workload optimization and self-healing in case of service-critical issues ([Gartner Research, 2019]). The main purpose of RCA is proactive detection, identification and resolution of upcoming infrastructure and application problems (in particular, performance degradations) before they negatively affect customers' environments. The main requirement from RCA is the low response time of a problem remediation with minimum resource utilization. The latest is especially important for cloud environments with larger restrictions on resources like CPU, GPU, memory, etc. compared to on-premises implementations. However, the most expected and valuable characteristic is the explainability of the outcome of RCA. Explainable AI (XAI) (see [Barredo Arrieta et al., 2020]) builds a trust towards the recommendations and increases the expertise for better system administration.

The traditional approach to management of cloud environments is monitoring of system-indicators like time series, logs and traces (known as three pillars of observability) for analysis, visualization, and reporting (see [Gartner Research, 2020a, Gartner Research, 2020b]). Self-driving data centers require the availability of proactive and intelligent analytics in view of nowadays very large-scale cloud environments. One of the important elements of the analytics is RCA on all acquired data for an anomaly detection, problem identification and self-remediation, risk management, intrusion detection, etc. (see [Mirgorodskiy et al., 2006, Mi et al., 2012, Marvasti et al., 2013, Agarwal and Agrawal, 2014, Kostroš et al., 2014, Wang et al., 2015, Sauvanaud et al., 2016, Tak et al., 2016, Solé et al., 2017, Neuvirth et al., 2015, Josefsson, 2017, Poghosyan et al., 2016, Harutyunyan et al., 2014, Harutyunyan et al., 2018b, Harutyunyan et al., 2020b, Chen et al., 2020, Das et al., 2020, Ferreira et al., 2020, Salaün et al., 2020, Bonandrini et al., 2020, Marcia et al., 2021, Jedrzej et al., 2019, Bilal et al., 2021, Vieira et al., 2021, Mejía et al., 2021, Ibrahim Kure and Islam, 2019, Poghosyan et al., 2021] with references therein).

Accumulated historical experience, while tackling similar problems, potentially can support proactive prediction of well-known issues ([Schnepp et al., 2017]). Ideally, the purpose of RCA should be identification of the main causes with assignment to proper administrators that will lead to accelerated remediation. Unfortunately, this ideal situation with the main roots within our control rarely realizes in practice. Normally, a problem can be identified and resolved only by controlling (monitoring) and managing the accompanying symptoms. Many RCA vendors like VMware (see [Harutyunyan et al., 2020a, Poghosyan et al., 2020, Marvasti et al., 2013]), BigPanda (see [BigPanda,

2020]) and Moogsoft (see [Sahil K., 2016]) follow this practical guidance describing a problem by its group of symptoms (alarms, alerts, events, etc.) within time and topology proximity named as an incident (episode or fingerprint). According to Moogsoft, an incident is a Probable Root Cause of a problem assuming that the surrounding symptoms possibly will help with a problem identification and resolution. According to BigPanda, an incident represents a high-level issue in an environment. They are created automatically by grouping together related alerts from the monitoring tools.

The flowchart of Figure 1 shows the classical approach to an incident detection and its application to a problem identification and accelerated remediation. The crucial role is given to alerts which should be defined by an expert or experienced user with sufficient knowledge regarding an environment, application and infrastructure key performance indicators, monitored data peculiarities, correlations and thresholds. They should be enough accurate to reduce the number of false positives and negatives.



Figure 1: The common approach to RCA based on alert definitions.

The process of Figure 1 might be rather complicated for realization in large-scale distributed environments with inconsistent behavior. The alert-based incident detection arises many questions as there is no common and straightforward approach. However, if those incidents are known with sufficient historical evidence, then the history of their resolutions will help to attach some annotations and store in the knowledge-base of historical patterns. In a run-time mode, a pattern matching process will compare the default group of alerts with the incidents of the knowledge-base and perform identification of the current status of an application or infrastructure. For a detected incident, the corresponding annotations will clarify its remediation strategies. Hence, the efficiency of the common RCA is entirely depending on the completeness of the alert space and on the ability of an incident generation. This method is very powerful and has a high-level explainability for many applications.

In this paper, our intention is the reinforcement of the common RCA (see Figure 1) by the automation of the incident generation process (see Figure 2). The process consists of data self-labelling and rule induction sub-processes. It works independently of alerts and directly utilizes monitoring datasets via machine learning algorithms. Figure 2 shows that rule-induction methods are responsible for an incident detection. Those methods can be

applied both to regression and classification problems where a preliminary data labelling is needed. The focus of our paper is application of classification rule-learner known as RIPPER (Repeated Incremental Pruning to Produce Error Reduction) (see [Cohen, 1995]). We consider data self-labelling via outlier detection as a system potential anomalies. The corresponding rules with sufficient statistical evidence explain/predict those outliers and serve as historical incidents. Incidents can be stored in the knowledge-base for further utilization. In the run-time mode, the RCA can perform pattern matching between monitoring datasets (without labelling) and the incidents (rules) of the knowledge-base for a problem identification.

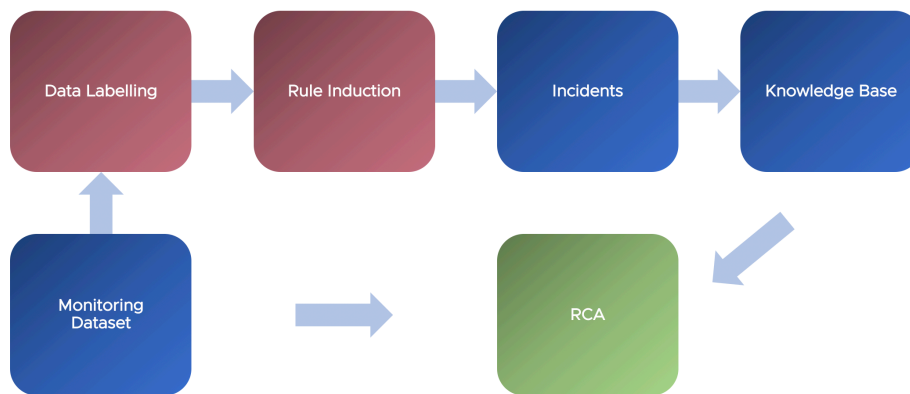


Figure 2: Intelligent RCA with data self-labelling and rule induction.

Paper [Poghosyan et al., 2020] realizes the procedure of Figure 2 via decision trees. It considers application of principal component analysis (PCA) (see [Jolliffe, 2002]) to the original monitoring dataset for dimensionality reduction and important/independent feature extraction. Principal components are uncorrelated and their simultaneous deviation from normality can be indication of a system anomaly behavior. Those deviations can be revealed by anomaly/outlier detection approaches. The paper applies the modification of the k-means clustering known as k-mod ([Chawla and Gionis, 2013]) to the space of principal components for outlier detection. Then, decision trees for classification are applied for prediction/explanation of the outliers. Each subtree predicting the outlier can be reformulated as a rule. The "strong" rules should be considered as incidents.

Decision trees have several disadvantages like sensitivity to noise, low predictive power and many others. The current paper considers application of more powerful rule-learning methods for classification problems. One of the most powerful rule-induction systems is RIPPER. It has several benefits like scalability, flexibility and explainability. RIPPER detects patterns in data as IF-THEN rules. The body of a rule (see [Fürnkranz and Kliegr, 2015]) consists of conditions each one containing a constraint that needs to be satisfied. The rule is said to fire if all conditions are satisfied and an instance is said to be covered by the rule. The rule head consists of a single class value (an outlier or a normal data point), which is predicted in case the rule fires (see [Fürnkranz and Kliegr, 2015]). The RIPPER rules can be simple (with only one condition) or complex (consisting of multiple conditions). The importance of a rule can be characterized by the

following two measures:

$$Confidence = \frac{N(body \rightarrow head)}{N(body)}, \quad (1)$$

and

$$Coverage = \frac{N(body \rightarrow head)}{N(positive\ class)}, \quad (2)$$

where $N(body \rightarrow head)$ is the number of instances that satisfy both the body and head of the rule, $N(body)$ is the number of instances that satisfy the body of the rule and $N(positive\ class)$ is the number of outliers. For an incident management, we need rules with high confidence (pure rules) while the coverage can be small for rare issues and large for more frequent ones. Collection of such pure rules makes our store of incidents.

In many applications/infrastructures the labelling of monitoring datasets can be performed via available key performance indicators (KPI-metrics). For example, application response time metric can be such an indicator. Then, there can be two different scenarios. The first one is data labelling based on those KPI-metrics, for example via some known thresholds with further application of classification rule-induction methods. The second approach can be direct application of regression rule-induction methods (like C4.5 or C5.0 regression methods) that can explain the values of the KPI-metrics through other indicators or capacity metrics. Finally, it is very important incorporation of users/experts knowledge into filtering of useless incidents and labels for overall improvement of the RCA. Moreover, incidents can be personalized for different users subject to their roles like application administrator, infrastructure administrator, security manager, etc..

The paper is organized as follows. Section 2 presents related works regarding RCA, outlier detection, rule-learners and their applications. Section 3 gives more detailed description of our approach. Section 4 considers experimental results for an application server. It discusses rule generation via principal components, monitored metrics and data labelling scenarios. Section 5 makes conclusions and gives ideas for future works.

2 Related work

One of the most important capabilities of the AI-powered data-analytics for IT operations (AIOps) is fully automated RCA ([Gartner Research, 2019]). Many AIOps platform vendors like IBM (see [IBM, 2021]), Facebook (see [Lin et al., 2020]), VMware (see [Marvasti et al., 2013, Marvasti et al., 2014a, Marvasti et al., 2014b, Marvasti et al., 2016, Marvasti et al., 2018, Poghosyan et al., 2016, Poghosyan et al., 2019a, Poghosyan et al., 2019b, Harutyunyan et al., 2019a, Harutyunyan et al., 2020b, Harutyunyan et al., 2020c]), HPE (see [HPE, 2019]), BigPanda (see [BigPanda, 2020]), DataDog (see [Othmane A.-A., 2021]), Moogsoft (see [Sahil K., 2016]) and others ([Moogsoft, 2016]) have almost complete vision and solution for the domain-centric RCA described in Figure 1.

Large-scale distributed cloud environments require intelligent RCA for self-driving data centers. It has been in the focus of researchers for decades with diverse ideas including anomaly detection, event correlations, causal inference, correlation analysis, predictive models and many others (see [ABS Consulting et al., 2014, Zawawy et al., 2010, Chuah et al., 2010, Cai et al., 2019, Marvasti et al., 2014b, Poghosyan et al., 2020] with references therein). Ideally, RCA should analyze all acquired monitoring datasets

including logs (see [Poghosyan et al., 2020, Harutyunyan et al., 2019b, Harutyunyan et al., 2018a, Harutyunyan et al., 2014, Mi et al., 2012, Kostroš et al., 2014, Tak et al., 2016, Chuah et al., 2010, Bird et al., 2015, Zawawy et al., 2010, Michalski, 1983]), traces (see [Suriadi et al., 2013, Lin et al., 2020]) and time series data (see [Jeyakumar et al., 2019, Pearl, 2009, Spirtes et al., 2000]) with possible correlations among them. Distributed tracing is the classical approach to application monitoring and diagnostics (see [Opentracing, 2019]). It is the best-known approach to the RCA of distributed systems.

Explainability of RCA improves the satisfaction with the recommendations and lead to more actionable insights. One of the powerful approaches to explainability is utilization of incidents with sufficient historical evidence ([Cohen et al., 2005, Bodik, 2010]). In this paper, we consider rule induction approaches to incident discovery. Rule-learners are the best if simplicity and explainability are important ([Fürnkranz, 1997, Fürnkranz, 1999, Fürnkranz et al., 2012, Fürnkranz and Kliegr, 2015, Agrawal et al., 1993, Galárraga et al., 2013, Quinlan, 2014, Clark and Niblett, 1989, Clark and Boswell, 1991]). RIPPER (Repeated Incremental Pruning to Produce Error Reduction) was the first rule learning system that effectively countered the overfitting problem ([Cohen, 1995, Hühn and Hüllermeier, 2009]). It is based on several previous works, most notably is the incremental reduced error pruning (IREP) idea ([Fürnkranz and Widmer, 1994]). In several independent studies, RIPPER has proved to be among the most competitive rule learning algorithms available today. RIPPER is a classification rule induction approach and the crucial steps for its realization is data labelling and feature selection/extraction in case of very large datasets. It is still the state-of-the-art in inductive rule learning with several important features like supporting missing values, numerical and categorical variables and multi-class classification problems.

Rule learning systems have wide applications including analysis of log-data and trace-data. Papers ([Suriadi et al., 2013, Lin et al., 2020]) consider analysis of log-data. Paper [Suriadi et al., 2013] compares RIPPER to C4.5 ([Quinlan, 2014]) for RCA on log data. It uses implementation of both algorithms in WEKA ([Witten et al., 2005]) known as JRip and J48 for RIPPER and C4.5, respectively. Paper ([Lin et al., 2020]) considers RCA in a large-scale production environment based on structured logs. It explores application of the Apriori algorithm ([Agrawal et al., 1993]) with subsequent improvement with FP-Growth ([Han et al., 2004]).

Papers ([Lee and Stolfo, 1998, Helmer et al., 2002, Helmer et al., 1998]) consider the problem of intrusion and abuse detection in computer systems in networked environments based on application traces. Paper ([Lee and Stolfo, 1998]) applies RIPPER to anomaly predictions. It also considers application of collective classifiers like association rules ([Höppner, 2005]) and frequent episodes algorithm ([Wu et al., 2012]) for improvement of the effectiveness in detecting intrusions. Paper ([Helmer et al., 2002, Helmer et al., 1998]) proposes a feature vector construction technique for system call-traces with further application of RIPPER for an anomaly classification. They also discussed a complexity reduction approach for learning algorithms.

Accurate and fast anomaly/outlier detection leads to proactive problem resolution and remediation before it affects a customer environment. Historical anomalies with their resolution experience are a valuable source for intelligent data labelling. Anomaly detection has been investigated by numerous authors for many applications (see [Thudumu et al., 2020, Blázquez-García et al., 2020, Pang et al., 2020, He et al., 2020, Chandola et al., 2009, Hodge and Austin, 2004, Zhang et al., 2020, Burgueño et al., 2020, Carta et al., 2020, Amarbayasgalan et al., 2020, Breunig et al., 2000, Chawla and Gionis, 2013, Marvasti et al., 2014a]). NN-based methods and deep learning are now becoming

very popular and powerful (see [Pang et al., 2020, Geiger et al., 2020] with references therein).

Another important step towards efficient RCA is data preprocessing via feature extraction or selection methods ([Cios et al., 2007, Ghoghgh et al., 2019, Jolliffe, 2002, John et al., 1994]). Naturally, the complexity is directly connected with the number of features involved in the learning. It should be very reasonable application of learning systems to only the important features. Feature subset selection has been shown to improve the performance of a learning algorithm and reduce the effort and amount of data required for machine learning on a broad range of problems ([Liu and Motoda, 1998]).

3 The main idea

We consider the scenario of Figure 2 for an incident generation, and Figure 3 describes the process in more details.

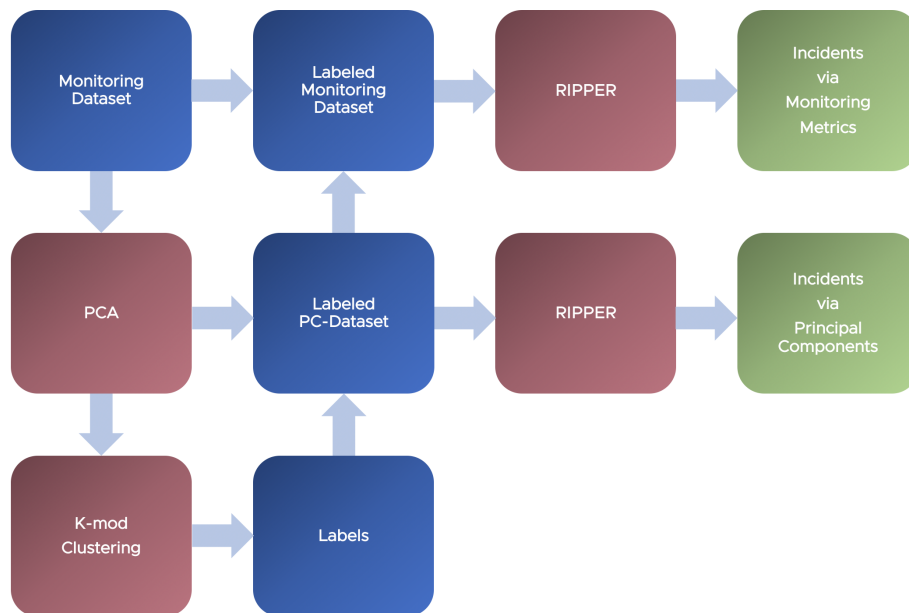


Figure 3: Incident generation via PCA, k-mod clustering and RIPPER.

RIPPER is selected for a rule construction. One of its main benefits compared to decision trees (see [Poghosyan et al., 2020]) is capability of automation of the rule generation process. RIPPER is a classification rule learner and preliminary data labelling is required. We suggest data self-labelling procedure in case if the time periods of performance degradations are unknown.

Data labelling consists of two steps: construction of principal components and detection of outliers. Application of PCA reduces the number of features for further analysis.

The final reduction rate of the original metric-space depends on a parameter for selection of the number of principal components that explain the required percentage of a dataset total variance. The value of the parameter can be within 90% – 99%. We select 95% for our experiments. The resultant principal components are uncorrelated which is important for outlier detection as simultaneous deviations of those metrics serve as a good indication of the system uncommon behavior. We perform outlier detection via k-mod approach ([Chawla and Gionis, 2013]). It utilizes k-means clustering with additional robustness to outliers.

Those labels can be attached to the original monitoring dataset with further application of RIPPER. In this case, the incidents/rules will be in terms of the original metrics. As we mentioned before, application of preliminary feature selection approaches should improve the quality of rules. This problem will be considered elsewhere. Those labels can be attached also to the dataset of principal components and RIPPER will generate rules based on those components. The latest approach has better performance due to smaller number of uncorrelated features compared to the original dataset, but worse explainability. Principal components as the linear combinations of the original metrics are difficult to interpret in the sense of application/infrastructure behavioral analysis. Further, the rules with large confidence measures can be stored as incidents. In our experiments, we used rules with confidences bigger than 90%.

In general, PCA is rather computationally complicated procedure. At least, in some implementations pairwise correlations should be calculated which can heavily impact the resources in case of big number of monitoring metrics. Hence, it is possible to perform data labelling directly based on the original metrics exactly by the same procedure. Both methods will coincide in case of small number of correlated metrics. Also, it is possible to apply feature selection or feature construction methods via other less complex approaches. Moreover, it is always beneficial to rely on an expert knowledge which can tremendously simplify the labelling process.

4 Results and discussions

We consider experimental realization of the approach described by Figure 3 and use real customer data from an application server. The monitoring dataset has 1290 features. The corresponding dataframe has 1290 columns and 44392 rows corresponding to 1-minute monitoring interval and almost 1-month data. We named the columns of the dataset as "V1", "V2", ..., "V1290". The exact names of the metrics are known and later will be provided.

We scale the original dataset by subtracting from each feature-column its average and dividing over its standard deviation. We apply PCA to the scaled dataset, and Figure 4 shows the percentage of variance explained by the first n principal components. We use the first 183 components for explaining 95% of the original variance. It provides with the 86% reduction of the original feature-space. We also see that for explaining the 99% of the variance, 300 principal components will be required with the compression rate of 77%.

Those 183 components compose a new PC-dataset. We reveal the outliers of the PC-dataset (the system possible anomalies) by application of k-mod algorithm designed for data clustering and outlier detection. It requires several parameters like the number of clusters and the percentage of possible outliers. We use the default setting $k = 5$ for the number of clusters and 7% for outliers (3000 data points). K-mod returns data points with outlying behavior which are labeled as anomaly class ("A" class) otherwise

as normal class ("N" class). Some of those outliers will be combined into rules while the others will remain as unexplained (miss-classified).

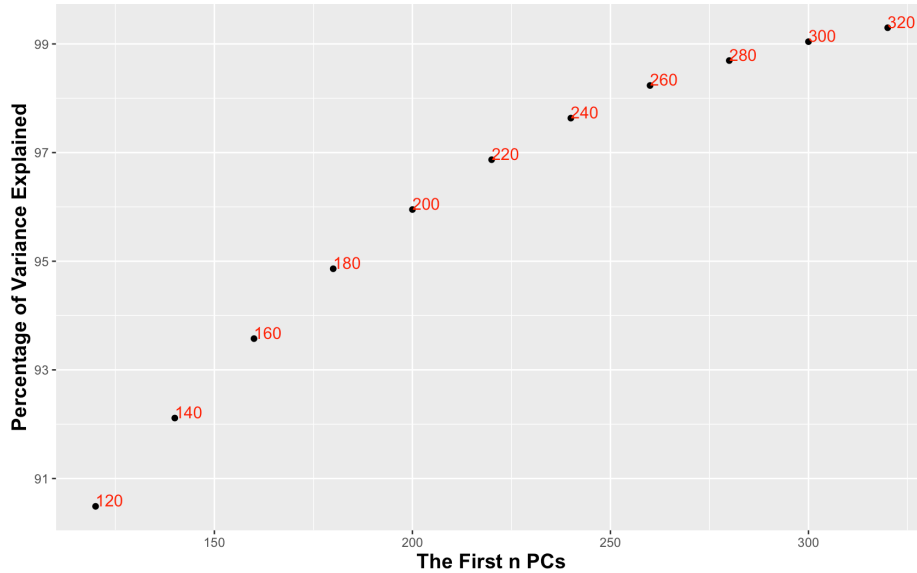


Figure 4: The percentage of variance explained by the principal components.

Figure 5 presents those outliers for the PC-dataset where the graph corresponds to the average of the absolute values of the first 183 principal components. Naturally, not all detected outliers visually match the correct ones as the 183-dimensional space is projected into the 1-dimensional average-metric space with distortion of some useful information. Other approaches for data labelling also should be considered elsewhere. Similarly, Figure 6 shows the detected outliers on the graph corresponding to the average of the absolute values of the scaled original metrics. We see that the time stamps corresponding to the outliers in the PC-space properly indicate outliers in the original metric-space. It is also connected with the properties of the principal components to preserve the required percentage of variance of the monitoring dataset.

In our experiments, we use implementation of RIPPER in WEKA library (see [Witten et al., 2005, Xu and Frank, 2020]). It is well-known under the name JRip (Java RIPPER) and was used by different authors for many similar problems. One of the important parameters of JRip (and RIPPER) is the number of optimizations. It is known as RIPPER k in case of k -step optimizations. The default is RIPPER2. We applied RIPPER10 for dramatic reduction of possible rules. Even in this case RIPPER provides with many rules that need further filtering based on the importance measures depending on the area of applications. We have selected the rules that have confidence higher than 90%. We don't use the "coverage" for historical incidents. In case of the run-time RCA both coverage and confidence should be large for a rule selection.

First, we apply RiPPER10 to the labeled PC-dataset. Figure 7 shows the output of JRip. The first line shows that we applied RIPPER10 (parameter "optimizations" = 10 indicating the number of optimization runs) and "minNo" = 50 indicating the total weight

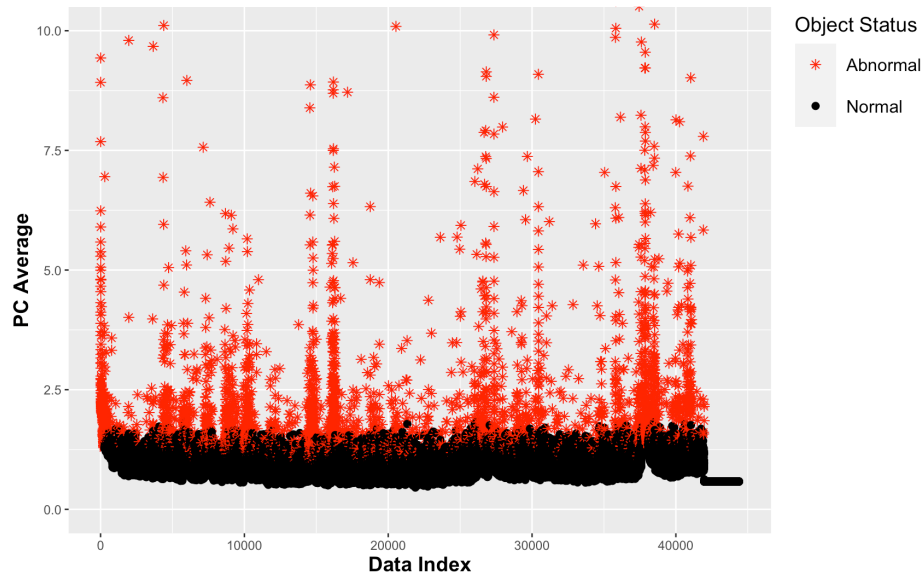


Figure 5: Outlier detection via PCA and k-mod clustering.

of the instances in a rule (see [Xu and Frank, 2020] for more details). The second and third lines show the dimensions of the PC-dataset together with the labels (named as "status"). RIPPER uses 10-fold cross validation for the testing. JRip reveals 17 rules. The first 16 rules explain anomaly labels (status = "A"). The fraction at the end of each rule indicates the number of instances that fire the rule (numerator) and the number of misclassifications (denominator). We calculate the confidence and coverage measures based on the fractions. The rules with high confidences are shown in Table 1 with the corresponding measures. We see both simple (with only one condition) and complex rules. They all predict/explain the "A" classes corresponding to outliers/anomalies (status = A). The execution time of JRip was 224 seconds.

Figure 8 shows the first 6 incidents of Table 1 across the time axis in different colors and shapes. We see incidents with different characteristics. Some of them are rare while the others are frequent. Some of the incidents contain bigger number of data points than the others. Additional expert knowledge might be very helpful for further rule filtering.

Second, we apply RIPPER10 to the original monitoring dataset with the labels derived as for the previous PC-dataset. The corresponding rules with confidences larger than 90% are shown in Table 2. The rules are in terms of the original metrics preliminary scaled before application of PCA, k-mod and RIPPER. The advantage of the rules in Table 2 is that experts can understand the nature of incidents and match them with real problems. The same matching process should be much difficult for the rules of Table 1. Figure 9 shows the incidents of Table 2 across the time axis. We see that the same outlying processes can be explained both via PC-rules and metric-rules. However, the best description remains unknown at this stage without proper system analysis.

We see from Figures 8 and 9 that measures like coverage and confidence (precision/recall) are not sufficient for further incident characterization. We need also to estimate the distribution of incidents across the time axis and duration of each one which will

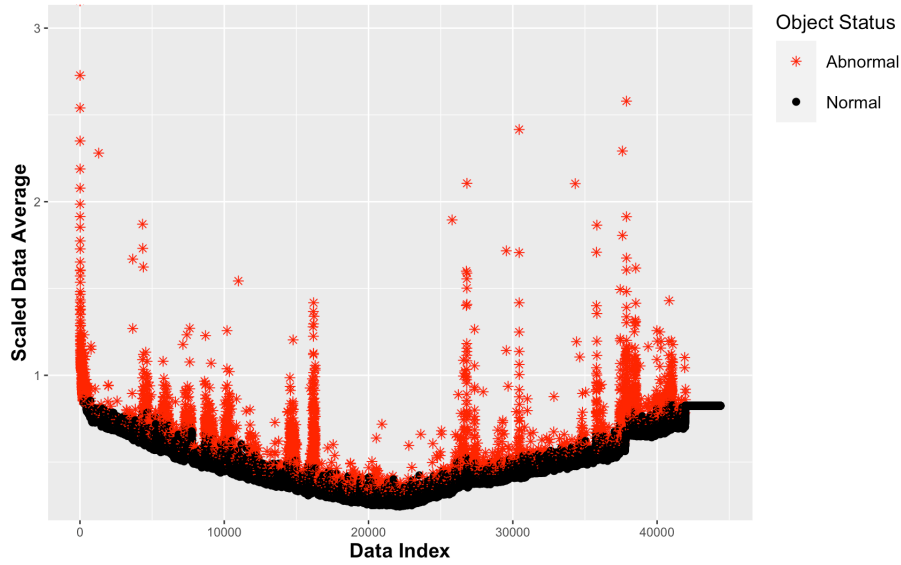


Figure 6: Outliers of Figure 5 shown for the original metric-space.

Incidents	Rules	Coverage	Confidence
1	$(PC6 \geq 8.08) \& (PC4 \leq -20.07)$	19.4%	98.98%
2	$PC11 \leq -7.45$	21.7%	92.4%
3	$(PC4 \leq -14.57) \& (PC6 \geq 16.64)$	1.9%	98.3%
4	$(PC24 \geq 3.90) \& (PC19 \geq 0.83)$	3.4%	97.2%
5	$(PC11 \leq -5.28) \& (PC2 \leq -16.88)$	1.8%	90.0%
6	$(PC6 \geq 7.42) \& (PC5 \geq -0.68) \& (PC41 \leq -1.03) \& (PC2 \leq 4.55) \& (PC1 \leq 19.88)$	2.5%	91.6%
7	$(PC126 \leq -0.95) \& (PC129 \geq 0.69) \& (PC131 \geq 1.57)$	3.23%	97.0%
8	$(PC24 \geq 1.66) \& (PC23 \leq -5.72) \& (PC113 \geq -0.25)$	2.0%	98.4%

Table 1: Incidents in terms of the principal components via RIPPER10.

help to match them properly with actual known problems. Detected incidents should pass further expert validation and scoring before storing in the knowledge base. Let us dig into the context of rules presented in Table 2. The third rule is the simplest one. The real name of "V21" metric is "alerting.copy-alert-latency.duration.max". The incident corresponds to Incident 3 of Figure 9. Incident 1 of Figure 9 is the most frequent. The real names of "V53", "V680" and "V844" are "alerting.latency.rate.m5", "fdb.tx.avrobase.get-with-ve", and "fdb.tx.bloom", respectively. Incident 4 is the most

Run information

Scheme: weka.classifiers.rules.JRip -F 3 -N 50.0 -O 10 -S 1
 Instances: 44392
 Attributes: 184
 Test mode: 10-fold cross-validation

JRIP rules:

(PC6 \geq 8.08)&(PC4 \leq -20.065) => status=A (588/6)
 (PC11 \leq -7.45) => status=A (707/54)
 (PC4 \leq -14.57)&(PC6 \geq 16.64) => status=A (59/1)
 (PC24 \geq 3.90)&(PC19 \geq 0.83) => status=A (106/3)
 (PC11 \leq -5.28)&(PC2 \leq -16.88) => status=A (60/6)
 (PC6 \geq 7.42)&(PC5 \geq -0.68)&(PC41 \leq -1.029)&(PC2 \leq 4.55)&(PC1 \leq 19.88) => status=A (83/7)
 (PC45 \leq -1.98) => status=A (127/32)
 (PC6 \geq 7.91)&(PC4 \leq -14.72)&(PC10 \leq 1.77) => status=A (159/46)
 (PC126 \leq -0.95)&(PC129 \geq 0.69) & (PC131 \geq 1.57) => status=A (100/3)
 (PC24 \geq 1.66)&(PC23 \leq -5.72)&(PC113 \geq -0.25) => status=A (62/1)
 (PC141 \leq -0.84)&(PC172 \leq -0.70)&(PC54 \geq -0.37) => status=A (91/20)
 (PC45 \leq -0.46)&(PC138 \geq 0.80)&(PC110 \leq 0.16) => status=A (143/69)
 (PC6 \geq 8.43)&(PC137 \geq 0.60) => status=A (174/86)
 (PC46 \geq 4.05) => status=A (110/31)
 (PC22 \geq 1.04)&(PC33 \geq 14.81) => status=A (86/20)
 (PC142 \leq -1.63) => status=A (174/84)
 => status=N (41563/640)

Number of Rules : 17

Time taken to build model: 223.7 seconds

Figure 7: JRip output.

rare in Figure 9. The real names of "V575" and "V578" are "fdb.tx.alert.alert-default-reindexer", and "fdb.tx.alertJob-box.advance.duration", respectively. Time series "V583" and "V576" are almost duplicates of the first two ones. It shows why preliminary feature selection should be highly important.

The process of data self-labelling can be simplified in case of available application or infrastructure performance indicators like workload or health that can directly specify anomalies. Time series outlier detection approaches can be applied to those indicators with the huge reduction of complexity. A powerful approach were considered in [Hyndman and Khandakar, 2008] based on time series STL decomposition (see [Cleveland et al., 1990, Hyndman and Athanasopoulos, 2018]) or smoothing methods. This method is implemented in R library as "tsoutliers" function. The case of key performance indicators will be considered elsewhere. Here, we deal with a synthetic indicators like the time series shown in previous figures. They are somehow identical to the application server workload.

We applied "tsoutliers" to the average of the absolute values of the first 183 principal components with detection of 1686 outliers presented in Figure 10. We can attach those labels both to the original dataset and to the PC-dataset. The RIPPER will reveal the rules in terms of the principal components or original metrics. Figure 11 shows some of the important rules found by RIPPER10 for the PC-dataset and labels from Figure 10.

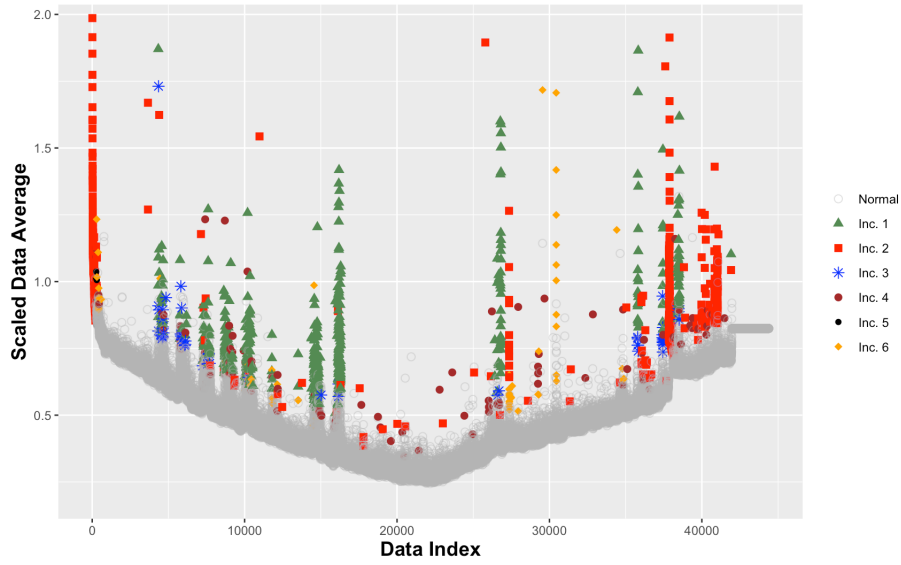


Figure 8: Incidents from Table 1 across the time axis.

Incidents	Rules	Coverage	Confidence
1	$(V53 \leq -0.63) \& (V680 \geq 0.33) \& (V844 \leq -1.7)$	14.1%	99.5%
2	$(V480 \geq -0.02) \& (V1123 \geq 2.53) \& (V739 \geq 2.18) \& (V1278 \leq -0.52)$	7.9%	97.5%
3	$V21 \geq 9.23$	3.2%	96.9%
4	$(V583 \geq -0.15) \& (V575 \geq 4.71) \& (V578 \geq 9.95) \& (V576 \geq 6.52)$	0.8%	92%

Table 2: Incidents in terms of the original metrics via RIPPER10.

We see that some of the anomalies are frequently spread across the time axis while others are rather rare and localized. The latest incidents should be more important. The graph in Figure 11 corresponds to the average of the absolute values of the scaled original metrics.

5 Conclusions and Future Work

We considered an approach to the root cause analysis of problems of cloud environments based on incidents with historical evidence. Historical evidence was indirectly revealed via rule induction systems. Detected incidents should be validated/scored by experts and stored in a knowledge-base with assigned annotations based on historical experience that led to accelerated or automated remediation of issues. The latests are one of the important components of the ML-powered data-analytics of self-driving data centers.

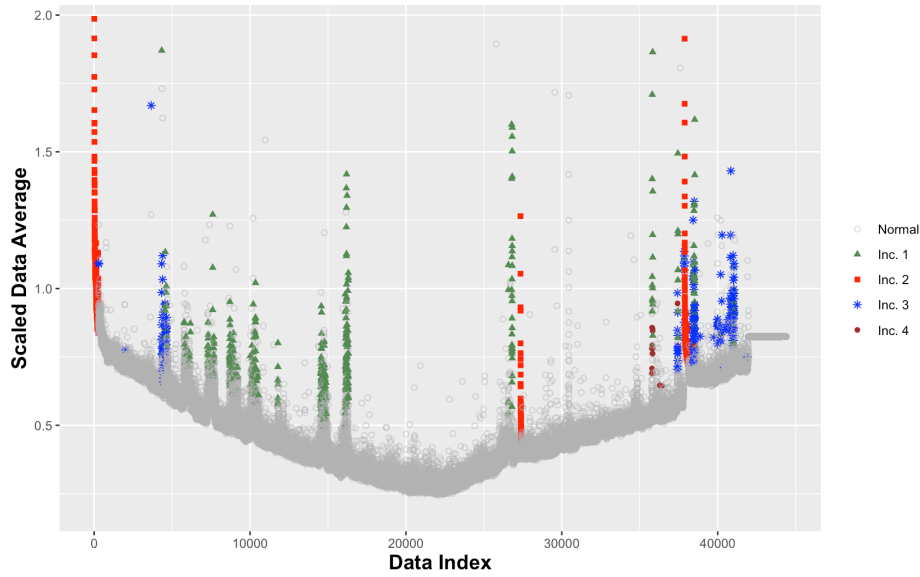


Figure 9: Incidents from Table 2 across the time axis.

The outcome of rule-induction methods is the list of rules for prediction or explanation of anomalies. The importance of rules can be estimated by several measures like coverage, confidence, and time frequency. The most important rules should be considered as incidents for further utilization. We used in our experiments the well-known method RIPPER as classification rule-learner. It has several benefits compared to other approaches and is the state-of-the-art in inductive rule learning. It can appropriately handle large noisy datasets. The corresponding significant rules serve as a problem identification with resolution recommendations. Interesting should be application of RIPPER to RCA for log-data and application traces.

Rule induction systems solve regression or classification problems and data labelling is the crucial step for the entire approach. We considered self-labelling approach based on combination of PCA and k-mod clustering. PCA reduces the dimensionality of the original monitoring space before application of the clustering. Moreover, it provides with uncorrelated principal components that preserve the variance and whose outlying behavior serves as an indication for an uncommon behavior. We considered application of RIPPER to the principal components and original metrics. The resultant rules can be either in terms of the principal components or the original metrics. The drawback of the first approach is in inexplicability of the principal components as application or infrastructure indicators. An interesting approach should be important feature selection before application of RIPPER to the original metric-dataset which will be considered elsewhere.

Overall, worth mentioning several important components of our approach. Labeled data is very rare and its derivation is connected with time and budget consuming procedures. Hence, the first benefit comes from self-supervised learning recommendations. In some applications semi-supervised approaches also can be applied. There is an ample of literature devoted to self-supervised learning ([Liu et al., 2021]) and semi-supervised

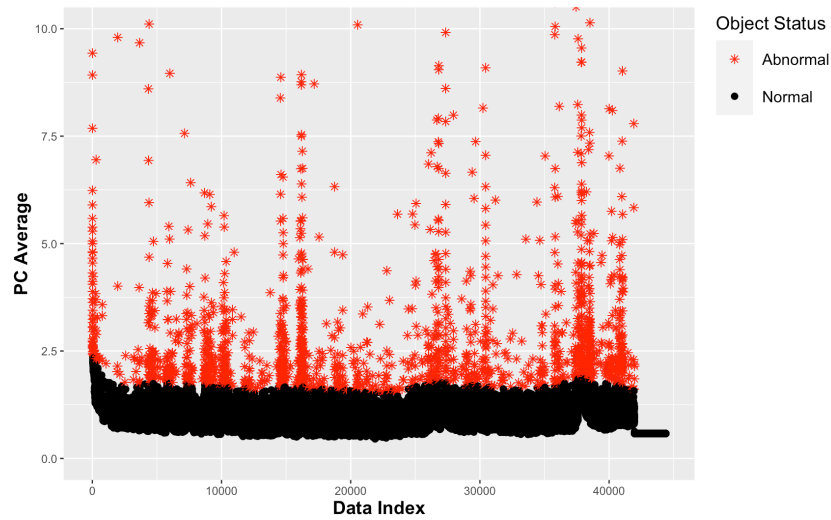


Figure 10: Outliers derived via "tsoutliers" function from R library.

learning ([Reddy et al., 2018]) problems and it should be discussed elsewhere for specific applications. The second benefit of our approach is the set of final rules that enhance the explainability of the RCA. They not only detect anomalies but explain and specify them. Explainability enhances our understanding of environments by uncovering different types of problems. Many of them can lead to proactive optimizations, restructuring and removal of potential bottlenecks.



Figure 11: Incidents via principal components with data labels of Figure 10.

Acknowledgements

Arnak Poghosyan was funded by RA Science Committee, in the frames of the research project № 20TTAT-AIa014.

We thank anonymous reviewers for careful and critical reading of our paper. Their insightful comments tremendously helped to improve our manuscript.

References

- [ABS Consulting et al., 2014] ABS Consulting, Vanden Heuvel, L., Lorenzo, D., Jackson, L., Hanson, W., Rooney, J., and Walker, D. (2014). *Root cause analysis handbook: A guide to efficient and effective incident investigation*. Rothstein Publishing.
- [Agarwal and Agrawal, 2014] Agarwal, P. and Agrawal, A. P. (2014). Fault-localization techniques for software systems: A literature review. *SIGSOFT Softw. Eng. Notes*, 39(5):1–8.
- [Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216.
- [Amarbayasgalan et al., 2020] Amarbayasgalan, T., Pham, V. H., Theera-Umpon, N., and Ryu, K. H. (2020). Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error. *Symmetry*, 12(8):1251.
- [Barredo Arrieta et al., 2020] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bannetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82 – 115.
- [BigPanda, 2020] BigPanda (2020). Incident management. <https://docs.bigpanda.io/docs/incident-management>. Accessed: 2021-01-26.
- [Bilal et al., 2021] Bilal, A.-S., Hamad, A., Basima, E., Tomayess, I., Pornpit, W., and Khadija, Khalid, P. (2021). Toward a knowledge-based personalised recommender system for mobile app development. *J. Univers. Comput. Sci. (JUCS)*, 27(2):208–229.
- [Bird et al., 2015] Bird, C., Menzies, T., and Zimmermann, T. (2015). *The art and science of analyzing software data*. Elsevier.
- [Blázquez-García et al., 2020] Blázquez-García, A., Conde, A., Mori, U., and Lozano, J. A. (2020). A review on outlier/anomaly detection in time series data. ArXiv:2002.04236v1.
- [Bodík, 2010] Bodík, P. (2010). *Automating data center operations using machine learning*. PhD thesis, University of California, Berkeley.
- [Bonandrini et al., 2020] Bonandrini, V., Bercher, J.-F., and Zangar, N. (2020). Machine learning methods for anomaly detection in iot networks, with illustrations. In Boumerdassi, S., Renault, É., and Mühlethaler, P., editors, *Machine Learning for Networking*, pages 287–295, Cham. Springer International Publishing.
- [Breunig et al., 2000] Breunig, M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104.
- [Burgueño et al., 2020] Burgueño, J., de-la Bandera, I., Mendoza, J., Palacios, D., Morillas, C., and Barco, R. (2020). Online anomaly detection system for mobile networks. *Sensors*, 20(24):7232.
- [Cai et al., 2019] Cai, Z., Li, W., Zhu, W., Liu, L., and Yang, B. (2019). A real-time trace-level root-cause diagnosis system in Alibaba datacenters. *IEEE Access*, 7:142692–142702.

- [Carta et al., 2020] Carta, S., Podda, A. S., Recupero, D. R., and Saia, R. (2020). A local feature engineering strategy to improve network anomaly detection. *Future Internet*, 12(10):177.
- [Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):1–58.
- [Chawla and Gionis, 2013] Chawla, S. and Gionis, A. (2013). k-means–: A unified approach to clustering and outlier detection. In *13th SIAM International Conference on Data Mining, Austin, Texas, 2013*, pages 189–197.
- [Chen et al., 2020] Chen, Y., Zhang, J., and Yeo, C. K. (2020). Network anomaly detection using federated deep autoencoding gaussian mixture model. In Boumerdassi, S., Renault, É., and Mühlethaler, P., editors, *Machine Learning for Networking*, pages 1–14, Cham. Springer International Publishing.
- [Chuah et al., 2010] Chuah, E., Kuo, S., Hiew, P., Tjhi, W., Lee, G., Hammond, J., Michalewicz, M. T., Hung, T., and Browne, J. C. (2010). Diagnosing the root-causes of failures from cluster log files. In *2010 International Conference on High Performance Computing (HiPC 2010)*, pages 1–10, Los Alamitos, CA, USA. IEEE Computer Society.
- [Cios et al., 2007] Cios, K. J., Swiniarski, R. W., Pedrycz, W., and Kurgan, L. A. (2007). *Feature extraction and selection methods*, pages 133–233. Springer US, Boston, MA.
- [Clark and Boswell, 1991] Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *European Working Session on Learning*, pages 151–163. Springer.
- [Clark and Niblett, 1989] Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine learning*, 3(4):261–283.
- [Cleveland et al., 1990] Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). Stl: A seasonal-trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6:3–73.
- [Cohen et al., 2005] Cohen, I., Zhang, S., Goldszmidt, M., Symons, J., Kelly, T., and Fox, A. (2005). Capturing, indexing, clustering, and retrieving system history.
- [Cohen, 1995] Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann.
- [Das et al., 2020] Das, A., Ajila, S. A., and Lung, C.-H. (2020). A comprehensive analysis of accuracies of machine learning algorithms for network intrusion detection. In Boumerdassi, S., Renault, É., and Mühlethaler, P., editors, *Machine Learning for Networking*, pages 40–57, Cham. Springer International Publishing.
- [Ferreira et al., 2020] Ferreira, D., Senna, C., Salvador, P., Cortesão, L., Pires, C., Pedro, R., and Sargento, S. (2020). Root cause analysis of reduced accessibility in 4g networks. In Boumerdassi, S., Renault, É., and Mühlethaler, P., editors, *Machine Learning for Networking*, pages 117–133, Cham. Springer International Publishing.
- [Fürnkranz, 1997] Fürnkranz, J. (1997). Pruning algorithms for rule learning. *Machine learning*, 27(2):139–172.
- [Fürnkranz, 1999] Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial intelligence review*, 13(1):3–54.
- [Fürnkranz et al., 2012] Fürnkranz, J., Gamberger, D., and Lavrač, N. (2012). *Foundations of rule learning*. Cognitive Technologies. Springer, Heidelberg. With a foreword by Geoffrey I. Webb.
- [Fürnkranz and Kliegr, 2015] Fürnkranz, J. and Kliegr, T. (2015). A brief overview of rule learning. In Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., and Roman, D., editors, *Rule technologies: Foundations, tools, and applications*, pages 54–69, Cham. Springer International Publishing.

- [Fürnkranz and Widmer, 1994] Fürnkranz, J. and Widmer, G. (1994). Incremental reduced error pruning. In Cohen, W. W. and Hirsh, H., editors, *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77, New Brunswick, NJ. Morgan Kaufmann.
- [Galárraga et al., 2013] Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. (2013). AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422.
- [Gartner Research, 2019] Gartner Research (2019). Market guide for AIOps platforms. <https://www.gartner.com/doc/reprints?id=1-1XS12Z80&ct=191118&st=sb>. Accessed: 2021-01-26.
- [Gartner Research, 2020a] Gartner Research (2020a). Application performance monitoring (APM) reviews and ratings. <https://www.gartner.com/reviews/market/application-performance-monitoring>. Accessed: 2021-01-26.
- [Gartner Research, 2020b] Gartner Research (2020b). It infrastructure monitoring tools. <https://www.gartner.com/reviews/market/it-infrastructure-monitoring-tools>. Accessed: 2021-01-26.
- [Geiger et al., 2020] Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., and Veeramachaneni, K. (2020). Tadgan: Time series anomaly detection using generative adversarial networks. ArXiv:2009.07769.
- [Ghojogh et al., 2019] Ghojogh, B., Samad, M. N., Mashhadi, S. A., Kapoor, T., Ali, W., Karray, F., and Crowley, M. (2019). Feature selection and feature extraction in pattern analysis: A literature review. ArXiv: 1905.02845.
- [Han et al., 2004] Han, J., Pei, J., and Yin, Y. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8:53–87.
- [Harutyunyan et al., 2019a] Harutyunyan, A. N., Grigoryan, N. M., and Poghosyan, A. V. (2019a). Methods and systems to enhance identifying service-level-objective degradation in a data center. Patent US 10181983 B2. Filed Jun 6, 2016. Issued Jan 15, 2019.
- [Harutyunyan et al., 2020a] Harutyunyan, A. N., Grigoryan, N. M., and Poghosyan, A. V. (2020a). Fingerprinting data center problems with association rules. In Hajian, A., Baloian, N., Inoue, T., and Luther, W., editors, *Proceedings of the Second CODASSCA Workshop, Yerevan, Armenia: Collaborative Technologies and Data Science in Artificial Intelligence Applications*, pages 152–158, Berlin. Logos Verlag.
- [Harutyunyan et al., 2020b] Harutyunyan, A. N., Grigoryan, N. M., Poghosyan, A. V., Dua, S., Antonyan, H., Aghajanyan, K., and Zhang, B. (2020b). Intelligent troubleshooting in data centers with mining evidence of performance problems. In Hajian, A., Baloian, N., Inoue, T., and Luther, W., editors, *Proceedings of the Second CODASSCA Workshop, Yerevan, Armenia: Collaborative Technologies and Data Science in Artificial Intelligence Applications*, pages 169–180, Berlin. Logos Verlag.
- [Harutyunyan et al., 2020c] Harutyunyan, A. N., Kushmerick, N., Poghosyan, A. V., Grigoryan, N. M., and Movsisyan, V. M. (2020c). Methods and systems to identify anomalous behaving components of a distributed computing system. Patent US 10572329 B2. Filed Dec 12, 2016. Issued Feb 25, 2020.
- [Harutyunyan et al., 2019b] Harutyunyan, A. N., Poghosyan, A. V., Grigoryan, N. M., Hovhannisyan, N. A., and Kushmerick, N. (2019b). On machine learning approaches for automated log management. *J. Univers. Comput. Sci. (JUCS)*, 25(8):925–945.
- [Harutyunyan et al., 2018a] Harutyunyan, A. N., Poghosyan, A. V., Grigoryan, N. M., Kushmerick, N., and Beybutyan, H. (2018a). Identifying changed or sick resources from logs. In *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, Italy, September 3-7, 2018*, pages 86–91. IEEE.
- [Harutyunyan et al., 2014] Harutyunyan, A. N., Poghosyan, A. V., Grigoryan, N. M., and Marvasti, M. A. (2014). Abnormality analysis of streamed log data. In *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*, pages 1–7. IEEE.

- [Harutyunyan et al., 2018b] Harutyunyan, A. N., Poghosyan, A. V., Kushmerick, N., and Grigoryan, N. (2018b). Learning baseline models of log sources. In Hajian, A., Luther, W., and Vinck, A. J. H., editors, *Proceedings of the CODASSCA Workshop, Yerevan, Armenia: Collaborative Technologies and Data Science in Artificial Intelligence Applications*, pages 145–156, Berlin. Logos Verlag.
- [He et al., 2020] He, Q., Zheng, Y., Zhang, C., and Wang, H. (2020). MTAD-TF: Multivariate time series anomaly detection using the combination of temporal pattern and feature pattern. *Complexity*, 2020(Article ID 8846608):1–9.
- [Helmer et al., 1998] Helmer, G., Wong, J., Honavar, V., and Miller, L. (1998). Intelligent agents for intrusion detection. In *Proceedings IEEE Information Technology Conference, Syracuse, NY*, pages 121–124. Springer.
- [Helmer et al., 2002] Helmer, G., Wong, J. S., Honavar, V., and Miller, L. (2002). Automated discovery of concise predictive rules for intrusion detection. *Information Fusion*, 60:165 – 175.
- [Hodge and Austin, 2004] Hodge, V. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.
- [Höppner, 2005] Höppner, F. (2005). Association rules. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*, pages 353–376. Springer US, Boston, MA.
- [HPE, 2019] HPE (2019). HPE InfoSight: Artificial intelligence for autonomous infrastructure. <https://h20195.www2.hpe.com/v2/getpdf.aspx/a00043401enw>. Accessed: 2021-01-26.
- [Hühn and Hüllermeier, 2009] Hühn, J. and Hüllermeier, E. (2009). FURIA: An algorithm for unordered fuzzy rule induction. *Data Min. Knowl. Discov.*, 19(3):293–319.
- [Hyndman and Athanasopoulos, 2018] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts: Melbourne, Australia.
- [Hyndman and Khandakar, 2008] Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software, Articles*, 27(3):1–22.
- [IBM, 2021] IBM (2021). RCA plugin. https://www.ibm.com/support/knowledgecenter/SSSHRK_4.2.0/event/concept/evnt_rootcauseanalysis.htm. Accessed: 2021-01-26.
- [Ibrahim Kure and Islam, 2019] Ibrahim Kure, H. and Islam, S. (2019). Cyber threat intelligence for improving cybersecurity and risk management in critical infrastructure. *J. Univers. Comput. Sci. (JUCS)*, 25(11):1478–1502.
- [Jedrzej et al., 2019] Jedrzej, B., Monika, S., Artur, J., and Krzysztof, S. (2019). Mobile agents for detecting network attacks using timing covert channels. *J. Univers. Comput. Sci. (JUCS)*, 25(9):1109–1130.
- [Jeyakumar et al., 2019] Jeyakumar, V., Madani, O., Parandeh, A., Kulshreshtha, A., Zeng, W., and Yadav, N. (2019). Explainit! – a declarative root-cause analysis engine for time series data. *Proceedings of the 2019 International Conference on Management of Data*.
- [John et al., 1994] John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the 11th International Conference*, pages 121–129. Morgan Kaufmann.
- [Jolliffe, 2002] Jolliffe, I. T. (2002). *Principal component analysis*. Springer series in Statistics. Springer-Verlag, New York, second edition.
- [Josefsson, 2017] Josefsson, T. (2017). *Root-cause analysis through machine learning in the cloud*. Uppsala Universitet.
- [Kostroš et al., 2014] Kostroš, M., Jakab, F., and Janitor, J. (2014). Overview of Big Data analysis for root cause determination and problem predictions. In *2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 249–254.

- [Lee and Stolfo, 1998] Lee, W. and Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, page 6, USA. USENIX Association.
- [Lin et al., 2020] Lin, F., Muzumdar, K., Laptev, N. P., Curelea, M.-V., Lee, S., and Sankar, S. (2020). Fast dimensional analysis for root cause investigation in a large-scale service environment. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2):1–23.
- [Liu and Motoda, 1998] Liu, H. and Motoda, H. (1998). *Feature extraction, construction and selection: A data mining perspective*. Springer.
- [Liu et al., 2021] Liu, X., Zhang, F., Hou, Z., Wang, Z., Mian, L., Zhang, J., and Tang, J. (2021). Self-supervised learning: Generative or contrastive.
- [Marcia et al., 2021] Marcia, H., Eulanda, S., Eduardo, S., and Altair, S. (2021). Spam detection based on feature evolution to deal with concept drift. *J. Univers. Comput. Sci. (JUCS)*, 27(4):364–386.
- [Marvasti et al., 2018] Marvasti, M. A., Harutyunyan, A. N., Grigoryan, N. M., and Poghosyan, A. V. (2018). Methods and systems to manage Big Data in cloud-computing infrastructures. Patent US 9948528 B2. Filed Apr 30, 2015. Issued Apr 17, 2018.
- [Marvasti et al., 2013] Marvasti, M. A., Poghosyan, A. V., Harutyunyan, A. N., and Grigoryan, N. M. (2013). Pattern detection in unstructured data: An experience for a virtualized IT infrastructure. In Turck, F. D., Diao, Y., Hong, C. S., Medhi, D., and Sadre, R., editors, *2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013, Ghent, Belgium, May 27-31, 2013*, pages 1048–1053. IEEE.
- [Marvasti et al., 2014a] Marvasti, M. A., Poghosyan, A. V., Harutyunyan, A. N., and Grigoryan, N. M. (2014a). An enterprise dynamic thresholding system. In Zhu, X., Casale, G., and Gu, X., editors, *11th International Conference on Autonomic Computing, ICAC 2014, Philadelphia, PA, USA, June 18-20, 2014*, pages 129–135. USENIX Association.
- [Marvasti et al., 2014b] Marvasti, M. A., Poghosyan, A. V., Harutyunyan, A. N., and Grigoryan, N. M. (2014b). Method and apparatus for root cause and critical pattern prediction using virtual directed graphs. Patent US 8751867 B2. Filed Oct 12, 2013. Issued Jun 10, 2014.
- [Marvasti et al., 2016] Marvasti, M. A., Poghosyan, A. V., Harutyunyan, A. N., and Grigoryan, N. M. (2016). Methods and systems for abnormality analysis of streamed log data. Patent US 9298538 B2. Filed Aug 6, 2013. Issued Mar 29, 2016.
- [Mejía et al., 2021] Mejía, J., Valencia-García, R., Alor-Hernández, G., and Calvo-Manzano, J. A. (2021). Knowledge intensive software engineering applications. *J. Univers. Comput. Sci. (JUCS)*, 27(2):87–90.
- [Mi et al., 2012] Mi, H., Wang, H., Zhou, Y., Lyu, M. R., and Cai, H. (2012). Localizing root causes of performance anomalies in cloud computing systems by analyzing request trace logs. *SCIENCE CHINA Information Sciences*, 55(12):2757–2773.
- [Michalski, 1983] Michalski, R. S. (1983). A theory and methodology of inductive learning. In *Machine Learning*, pages 83–134. Elsevier.
- [Mirgorodskiy et al., 2006] Mirgorodskiy, A. V., Maruyama, N., and Miller, B. P. (2006). Problem diagnosis in large-scale computing environments. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, pages 11–11.
- [Moogsoft, 2016] Moogsoft (2016). Moogsoft alternatives and competitors. <https://www.g2.com/products/moogsoft/competitors/alternatives>. Accessed: 2021-01-26.
- [Neuvirth et al., 2015] Neuvirth, H., Finkelstein, Y., Hilbuch, A., Nahum, S., Alon, D., and Yom-Tov, E. (2015). Early detection of fraud storms in the cloud. In Bifet, A., May, M., Zadrozny, B., Gavalda, R., Pedreschi, D., Bonchi, F., Cardoso, J., and Spiliopoulou, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 53–67, Cham. Springer International Publishing.

- [Opentracing, 2019] Opentracing (2019). What is distributed tracing? <https://opentracing.io/docs/overview/what-is-tracing/>. Accessed: 2021-01-26.
- [Othmane A.-A., 2021] Othmane A.-A. (2021). Automated root cause analysis with Watchdog RCA. <https://www.datadoghq.com/blog/>. Accessed: 2021-01-26.
- [Pang et al., 2020] Pang, G., Shen, C., Cao, L., and van den Hengel, A. (2020). Deep learning for anomaly detection: A review. ArXiv:2007.02500.
- [Pearl, 2009] Pearl, J. (2009). *Causality*. Cambridge University Press.
- [Poghosyan et al., 2021] Poghosyan, A., Harutyunyan, A., Grigoryan, N., Pang, C., Oganessian, G., Ghazaryan, S., and Hovhannisyan, N. (2021). An enterprise time series forecasting system for cloud applications using transfer learning. *Sensors*, 21(5).
- [Poghosyan et al., 2016] Poghosyan, A. V., Harutyunyan, A. N., and Grigoryan, N. M. (2016). Managing cloud infrastructures by a multi-layer data analytics. In Kounev, S., Giese, H., and Liu, J., editors, *2016 IEEE International Conference on Autonomic Computing, ICAC 2016, Wuerzburg, Germany, July 17-22, 2016*, pages 351–356. IEEE Computer Society.
- [Poghosyan et al., 2020] Poghosyan, A. V., Harutyunyan, A. N., Grigoryan, N. M., and Kushmerick, N. (2020). Learning data center incidents for automated root cause analysis. In Hajian, A., Baloiian, N., Inoue, T., and Luther, W., editors, *Proceedings of the Second CODASSCA Workshop, Yerevan, Armenia: Collaborative Technologies and Data Science in Artificial Intelligence Applications*, pages 181–190, Berlin. Logos Verlag.
- [Poghosyan et al., 2019a] Poghosyan, A. V., Harutyunyan, A. N., Grigoryan, N. M., and Marvasti, M. A. (2019a). Data-agnostic anomaly detection. Patent US 10241887 B2. Filed Mar 29, 2013. Issued Mar 26, 2019.
- [Poghosyan et al., 2019b] Poghosyan, A. V., Harutyunyan, A. N., Grigoryan, N. M., and Marvasti, M. A. (2019b). Methods and systems to determine baseline event-type distributions of event sources and detect changes in behavior of event sources. Patent US 10509712 B2. Filed Nov 30, 2017. Issued Dec 17, 2019.
- [Quinlan, 2014] Quinlan, J. R. (2014). *C4.5: programs for machine learning*. Elsevier.
- [Reddy et al., 2018] Reddy, Y. C. A. P., Viswanath, P., and Reddy, B. E. (2018). Semi-supervised learning: a brief review. *International Journal of Engineering and Technology*, 7(1.8):81–85.
- [Sahil K., 2016] Sahil K. (2016). A journey through it incident management. <https://www.moogsoft.com/blog/aiops/journey-through-incident-management/>. Accessed: 2021-01-26.
- [Salaün et al., 2020] Salaün, A., Bouillard, A., and Buob, M.-O. (2020). Space-time pattern extraction in alarm logs for network diagnosis. In Boumerdassi, S., Renault, É., and Mühlethaler, P., editors, *Machine Learning for Networking*, pages 134–153, Cham. Springer International Publishing.
- [Sauvanaud et al., 2016] Sauvanaud, C., Lazri, K., Kaâniche, M., and Kanoun, K. (2016). Anomaly detection and root cause localization in virtual network functions. In *27th International Symposium on Software Reliability Engineering (ISSRE 2016)*, Proceedings of the 27th International Symposium on Software Reliability Engineering (ISSRE 2016), pages 196 – 206, Ottawa, Canada.
- [Schnepp et al., 2017] Schnepp, R., Vidal, R., and Hawley, C. (2017). *Incident Management for operations*. O'Reilly Media, Inc.
- [Solé et al., 2017] Solé, M., Muntés-Mulero, V., Rana, A. I., and Estrada, G. (2017). Survey on models and techniques for root-cause analysis. ArXiv:1701.08546.
- [Spirtes et al., 2000] Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, prediction, and search*. MIT press, 2nd edition.

- [Suriadi et al., 2013] Suriadi, S., Ouyang, C., van der Aalst, W., and ter Hofstede, A. (2013). Root cause analysis with enriched process logs. In Rosa, M. L. and Soffer, P., editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2012)*. Volume 132 of *Lecture Notes in Business Information Processing*, pages 174–186, Berlin. Springer-Verlag.
- [Tak et al., 2016] Tak, B., Tao, S., Yang, L., Zhu, C., and Ruan, Y. (2016). LOGAN: Problem diagnosis in the cloud using log-based reference models. *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pages 62–67.
- [Thudumu et al., 2020] Thudumu, S., Branch, P., Jin, J., and Singh, J. J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional Big Data. *Journal of Big Data*, 7(42):43 – 57.
- [Vieira et al., 2021] Vieira, M. A., Ribeiro, E. L. F., Claro, D. B., and Mane, B. (2021). Integration model between heterogeneous data services in a cloud. *J. Univers. Comput. Sci. (JUCS)*, 27(4):387–412.
- [Wang et al., 2015] Wang, T., Zhang, W., Wei, J., and Zhong, H. (2015). Fault detection for cloud computing systems with correlation analysis. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 652–658.
- [Witten et al., 2005] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2005). Practical machine learning tools and techniques. *Morgan Kaufmann*.
- [Wu et al., 2012] Wu, J., Wan, L., and Xu, Z. (2012). Algorithms to discover complete frequent episodes in sequences. In Cao, L., Huang, J. Z., Bailey, J., Koh, Y. S., and Luo, J., editors, *New Frontiers in Applied Data Mining*, pages 267–278, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Xu and Frank, 2020] Xu, X. and Frank, E. (2020). Algorithm JRip in Weka. <https://weka.sourceforge.io/doc.dev/weka/classifiers/rules/JRip.html>. Accessed: 2021-01-10.
- [Zawawy et al., 2010] Zawawy, H., Kontogiannis, K., and Mylopoulos, J. (2010). Log filtering and interpretation for root cause analysis. In *2010 IEEE International Conference on Software Maintenance*, pages 1–5.
- [Zhang et al., 2020] Zhang, M., Guo, J., Li, X., and Jin, R. (2020). Data-driven anomaly detection approach for time-series streaming data. *Sensors*, 20(19):5646.