Journal of Universal Computer Science, vol. 29, no. 11 (2023), 1298-1318
submitted: 19/5/2023, accepted: 30/9/2023, appeared: 28/11/2023

Challenges and Experiences in Designing Interpretable KPI-diagnostics for Cloud Applications

Ashot Harutyunyan
(AI Lab at Yerevan State University, Yerevan, Armenia
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)

Arnak Poghosyan
(Institute of Mathematics of NAS RA, Yerevan, Armenia
VMware, Palo Alto, US
https://orcid.org/0000-0002-6037-4851, apoghosyan@vmware.com)

Lilit Harutyunyan
(VMware, Palo Alto, US
https://orcid.org/0000-0002-9558-9385, lharutyunyan@vmware.com)

Nelli Aghajanyan
(Deutsche Börse AG, Frankfurt am Main, Germany
https://orcid.org/0000-0002-3560-7502, nelli.aghajanyan@deutsche-boerse.com)

Tigran Bunarjyan
(Institute of Mathematics of NAS RA, Yerevan, Armenia
Technische Universität München, Munich, Germany
VMware, Palo Alto, US
https://orcid.org/0000-0003-4427-4284, tbunarjyan@vmware.com)

A.J. Han Vinck
(University of Duisburg-Essen, Duisburg, Germany
https://orcid.org/0000-0003-3437-3676, han.vinck@uni-due.de)

Abstract: Automated root cause analysis of performance problems in modern cloud computing infrastructures is of a high technology value in the self-driving context. Those systems are evolved into large scale and complex solutions which are core for running most of today's business applications. Hence, cloud management providers realize their mission through a "total" monitoring of data center flows thus enabling a full visibility into the cloud. Appropriate machine learning methods and software products rely on such observation data for real-time identification and remediation of potential sources of performance degradations in cloud operations to minimize their impacts. We describe the existing technology challenges and our experiences while working on designing problem root cause analysis mechanisms which are automatic, application agnostic, and, at the same time, interpretable for human operators to gain their trust. The paper focuses on diagnosis of cloud ecosystems through their Key Performance Indicators (KPI). Those indicators are utilized to build automatically labeled data sets and train explainable AI models for identifying conditions and processes "responsible" for misbehaviors. Our experiments on a large time series data set from a cloud application demonstrate that those approaches are effective in obtaining

models that explain unacceptable KPI behaviors and localize sources of issues.

Keywords: Cloud infrastructures, Key Performance Indicators (KPI), automated root cause analysis (RCA), explainable ML/AI, feature importance, rule induction
Categories: I.2, K.6
DOI: 10.3897/jucs.112570

1 Intelligent Diagnosis of Cloud Environments

1.1 Explaining Factors of Misbehaviors

The general approach currently used by companies when it comes to RCA of performance issues in the customer ecosystems and own products is to rely on expertise of on-site users or support engineers. Expert efforts and knowledge are not anymore adequate for reliable management and quick remediation of misbehaving components of modern cloud environments tending to build self-driving capabilities (such as an application KPI optimization, the vision behind project Magna, https://blogs.vmware.com/virtualblocks/2019/08/26/vsan-project-magna/). Backtracking and finding root causes of failures in those distributed environments with high degree of sophisticated interrelations between data center objects is an unrealistic manual task for human operators.

Machine learning helps to automate the management of such complex systems [Josefsson 2017], [Sole et al 2017] that contain thousands of objects like VMs, Hosts, datastores, via monitoring millions of time series metrics, terabytes of logs, and application traces, to capture a high-resolution "image" of the entire stack. However, self-diagnostics of issues with those intelligent monitoring and analytics solutions (cloud vendors' products) is another fundamental problem at the customer environments requiring time-intensive analysis of support experts and substantial long-term investments. VMware Skyline [VMware Skyline 2022] summarizes common patterns (with the field experts involved) of product problems into remediation "rules" for more proactive support at the customer site later. A closed-loop global rule learning from products usage and performance data to maintain product KPIs healthy would be the advanced path several other companies currently follow (see HPE InfoSight [HPE 2022]).

Although data center management market progresses towards AI Ops solutions, like VMware Area (former vRelaize Operations Manager) [VMware Aria 2022], it still is providing only semi-automated root cause detection capabilities for customer applications under supervision, as well as for self-diagnosis, although enriched with various intelligent troubleshooting toolsets. In particular, they make finding evidence of potential causes of an alert or data center situation easier with discovery of "interesting" changes occurring in system events space, configuration properties, and data center flows for further user validation and decision making. Such a troubleshooting analytics [Harutyunyan et al 2020(1)] may apply statistical change point detection methods and entropic measures to derive the ranked lists of relevant patterns according to their importance. This kind of unsupervised approaches mitigate RCA problem but also produce false positive noise and redundancy. Full automation of RCA of an issue or its prediction remains unresolved. Our prior works ([Harutyunyan et al 2020(1)], [Poghosyan et al 2020(1)], [Harutyunyan et al 2020(2)], [Bunarjyan et al 2020], [Poghosyan et al 2020(2)], [Harutyunyan et al 2022], [Poghosyan et al 2022], [Baghdasaryan et al 2022]) reported in CODASSCA 2020 and 2022 workshops, extended papers ([Poghosyan et al 2021(1)], [Harutyunyan et al 2019], [Poghosyan et al 2021(2)]) in J.UCS and Sensors special issues on those workshops,

earlier research ([Harutyunyan et al 2018], [Harutyunyan et al 2014]) on intelligent log analytics, represent various sorts of attempts at enabling and facilitating incident discovery and prediction, as well as RCA capabilities of cloud management solutions from different perspectives. In particular, the current paper builds on [Harutyunyan et al 2022].

Automated RCA with machine intelligence is a core problem in the self-driving data centers context [SDDC 2017]. However, for gaining user trust in ML solutions it is also essential and preferable to build such technologies on top of interpretable models [Barredo Arrieta et al 2020], [Fürnkranz et al 2012]. Core problems in reliable and intelligent cloud operations (including KPI diagnosis) are addressed in recent works ([Chen et al 2020], [Lyu et al 2021], [Lyu and Su 2023], [Wang et al 2023]) by various research groups.

There are multiple factors that hinder designing effective RCA solutions with ML for cloud computing infrastructures and applications, the main ones are

- lack or absence of labeled data;
- operator or expert verified/annotated/labeled data sets are hard to obtain in this domain and ungeneralizable from one environment to another because of ecosystem specifics.

Another aspect which is core to take into consideration is the explainability of an automated RCA. Industry is increasingly extending its frontiers with ML and AI, while facing the problem of explainability [Barredo Arrieta et al 2020] of sophisticated deep learning models and their outcomes. Although accurate ML models are valuable for automated RCA, their explainability is a desired feature to justify the reasons of failures and conditions that lead to such degradations. In addition to indicating actionable recommendations, those conditions are uncovering knowledge that can be leveraged in further optimization of the application. So, building trust between the user and AI should be an important requirement in designing data center diagnostics of the future. Therefore, our goal should be developing effective RCA methods which enable also explainable AI for products developed to manage cloud environments. That implies identifying ways to design intelligent systems that run on models with optimal trade-offs between their predictive power and explainability).

In that context, we outline ideas and a prototype solution for an automated RCA in terms of diagnosing KPI degradations that target troubleshooting customer data centers and/or cloud management products residing in their environments (thus enabling a proactive support while collecting high-frequency telemetry data from the products).

This paper presents some techniques and ML models that predict the potential causes of system's failures subject to its KPI, an underlying goal in the project Magna-Diagnosis mentioned above. Several regression and classification models were trained and analyzed using concepts of variable importance, decision trees and rule learners, as well as neural networks to identify and explain KPI degradations for a vRealize Operations deployment. These findings derived from high-accuracy ML models, which lack in human ground truth, were evaluated by the application developers to estimate their utility in practice and overall compliance with their expertise in long-term troubleshooting of the product issues.

Rule induction is always preferable if the interpretability of the models and patterns are required compared to their predictive power [Clark and Boswell 1998]. There are various rule induction algorithms [Fürnkranz et al 2012]. In this work we experimented with CN2 (see its implementation in the visual programming tool Orange [Orange 2023]).

In view of the above-mentioned challenges, our experience with designing relevant diagnostics pipeline proposal relies on the following building blocks:

1. Leveraging KPI metric as a source for generating labels for the entire data set of the application, while quantizing it into two or more class IDs;
2. Training regression and classification models to employ those in predicting KPI failures, while also evaluating relative variable/feature importance scores of those models to be utilized for explainability purposes;
3. Applying decision trees and rule induction (a form of explainable AI) algorithms to derive consistent conditions of KPI failures for a full KPI diagnosis and interpretability.

Methods are generic in nature and can be applied to different use cases, including proactive/predictive customer support for deployed cloud management instances or SaaS-based delivery of those services.

For an exemplary distributed application and its selected KPIs such as a latency metric, interpretable models are trained, validated against expertise of application developers, and used for producing run-time root cause recommendations on KPI abnormalities.

Overall, the objective of such a study is to identify important features and conditions of cloud applications subject to impact on KPIs. Based on this, intelligent cloud management solutions can provide recommender systems for optimizing applications performance (while indicating those important variables to be tuned) and predicting patterns causing unacceptable performance states (hence, accelerate the system recovery). This work focuses on experimental evaluation of the self-diagnostic use case of the technology leading cloud management solution VMware Aria Operations [VMware Aria 2022].

Figure 1 depicts this product application in its functions to monitor and guard multi-cloud infrastructures. The diagram reflects three cloud environments built on

1. VMware compute/storage/network virtualization solutions vSphere, vSAN, and NSX [VMware Prods 2023],
2. such an infrastructure hosted in Microsoft Azure, and
3. native Azure service, respectively.

While this distributed application is intelligently managing various cloud environments, the product itself might greatly benefit from self-healing capabilities or automated recommendations for performance improvements and recovery from misbehaviors. Architectural specifics and variety of workload patterns at different types of clouds may affect/stress vRealize Operations performance differently. Therefore, for self-diagnostics purposes, special models need to be trained for each case with specific requirements on KPIs behavior set by users.

Various ML algorithms are employed for comparative analysis including neural networks. Expert validation of discovered patterns promises wider adoptability of the approaches in real world scenarios with limited or unavailable annotated data sets.

1.2 Methodology Frameworks and Paper Sttructure

In our study we apply both regression and classification methods, including rule induction algorithms, as well as information-theoretic feature ranking techniques. In one scenario,

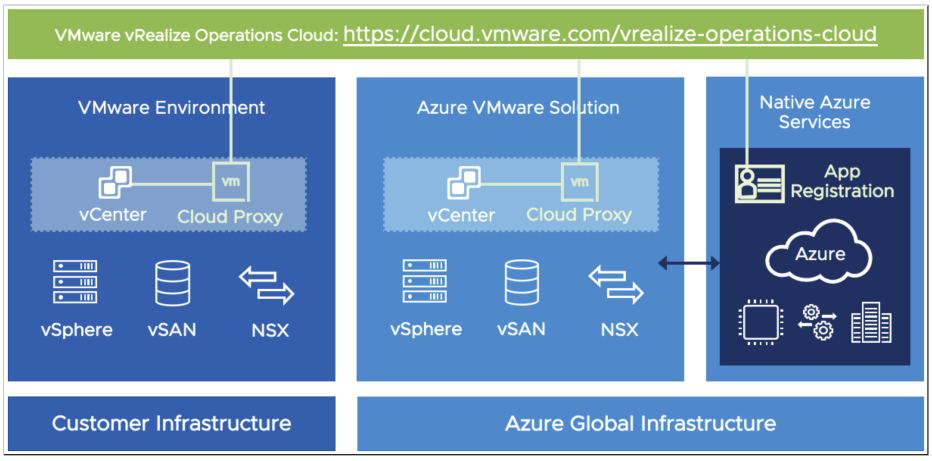

Figure 1: vRealize Operations in multi-cloud management.

we are interested in identifying potential factors explaining the KPI behavior, in another one, the problem is to predict KPI abnormality and deduce ”rules” leading to such situations.

Further research needs to address the problem of efficient management of multiple trade offing KPIs for an application.

The paper is organized as follows. Section 2 specifies use cases of application diagnostics from KPI perspectives and a sampled data set representing the performance monitoring of that application consisting of thousands of features. Section 3 focuses on experimental aspects of our investigations, while Section 4 expands on analysis of patterns. Section 5 shares validation results and challenges. Section 6 contains forward looking perspectives on this research and its productization with internal reviewer feedback on technology value.

2 Use Case of a Product KPI Diagnosis

We explain our methods on a data set measured by vRealize Operations (a multi-node distributed application consisting of several VMs) regarding its performance in terms of its self-monitoring metrics within a real deployment in one of our data centers. The product collects a large amount of such time series metrics that keep track of its own performance. They give a thorough picture of application’s state. The dataset we analyzed represents a collection of the application’s self-monitoring metrics for a node from a 6-node product deployment.

2.1 Dataset

The raw dataset was processed against gaps and normalized for further analysis. A node-specific data frame consisting of 3000 time series features (subject to 5 min regular monitoring interval) was considered, both numeric and categorical with 5100 instances each for a period of 18 days.

2.2 Key Performance Indicators

Different target variables (KPIs, see the relevant list [Self-Monitoring 2022] for vRealize Operations) in the dataset were used to generate labels for the rest of the data frame applying high quantile values of the metric. For training classification models, binary labels (normal vs abnormal state of the KPI) were generated using such an artificial and self-labeling technique. This way the dataset could be fed into classification algorithms for automated RCA to predict the positive/degraded class. The meaning behind this is to have ready ML models that could derive conditions/rules that interpret the degradation of KPI (positive class) or indicate the most influential dimensions/features for a long-term explainability of those degradations. To get better understanding of this approach, we have discretized the target variable to be 0 or 1 based on the 97, 95, 93, 90-quantiles of the KPI and picked up the quantile which resulted in the best performance.

New global KPIs have also been constructed since there are cases when a particular KPI cannot describe the required performance aspect accurately when taken separately. As node's behavior must be evaluated relative to the remaining nodes, taking a single KPI that reflects the relationship between only one to another node would not be descriptive. For each node from the 6-node product deployment, there are 5 self-monitoring metrics for each remaining node, which show the maximum/average response latency from the current node to others.

The super-metrics

Node|PingLatency|Max of Max

and

Node|PingLatency|Avg of Avg (Figure 2)

used for RCA are constructed by taking the maximum/average value among all 5 metrics' observations at a given point in time.

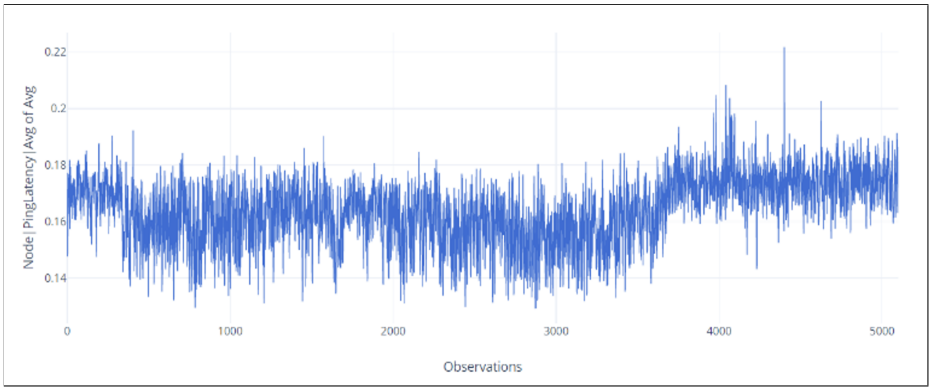

Figure 2: Plot for Node|PingLatency|Avg of Avg.

3 Experiments and Discussions

As mentioned above, we are interested in explaining a specific KPI degradation instance with classification and rule induction approaches, as well as in identifying flows highly influential on the KPI from historic perspectives (which implies employing regression settings). Both are important tasks. The first one explains an application situation, the second one derives factors that can be taken into account for application optimization planning. Moreover, highly important features and potential anomalies/outliers on them in a specific situation can also indicate reasons for a KPI failure.

In the regression analysis, a target variable (KPI) was chosen to be

Node|PingLatency|Max of Max.

k Nearest Neighbor (kNN) regression (to model expected non-linearity in the dataset) with parameter equal to 115 with cross-validation from the range $k = [5; 123]$ has yielded Root Mean Square Error (RMSE) equal to 0.119.

Another KPI was the constructed general KPI

Node|Ping Latency|Avg of Avg

shown in Figure 2. The kNN parameter was 15 with cross-validation from the range $k = [5; 123]$ and the algorithm has output RMSE equal to 0.062.

The third chosen KPI was

OverallThresholdChecking|MaxDuration,

which records the maximum duration for actions of those items that are used to process incoming observation data (against baseline thresholds for time series metrics set by the user or learned statistically). Here the number of neighbors was chosen to be 63 with cross-validation from the range $k = [5; 123]$ with RMSE equal to 0.027.

Table 1 summarizes top features subject to their relative importance values ("Coef" columns in the table) obtained for the kNN method and the corresponding KPIs (with short descriptors of those metrics as *Max of Max*, *Avg of Avg*, and *MaxDuration*, respectively). In terms of RMSE, the models performed well.

To benefit from supervised learning methods, we self-labeled the data frame with outlying behaviors of the KPI as positive class, using for that $97, 95, 93, 90-$quantiles of the metric in distinct experimental scenarios.

All above mentioned KPIs were considered to train different models (with several traditional classification algorithms) and compare their accuracies. However, the evaluation results for all those KPIs and ML algorithms were not satisfactory. The models did not have enough discrimination capacity to distinguish between positive and negative classes. This was a consequence of noisy dataset and class imbalance for the target variable. 90-quantile cut-off produced better results, so we considered this quantile for further improvement of the models.

We present a possible resolution to the problem of noise in the dataset below and the results obtained after improvements.

3.1 Neural Networks

Compared to traditional classification algorithms, Multi-Layer Perceptron (MLP) demonstrates a completely different predictive power on our data set. In particular, with 97-quantile cut-off for *Node|PingLatency|Avg of Avg*, MLP produces 96%-accuracy with

Precision=95% and Recall=0.96% in the binary classification between normal and abnormal states of the KPI. However, because of the interpretability issue of neural networks, we have to proceed with finding ways to improve our initial results on explainable models compromising the predictive power.

Max of Max		**Avg of Avg**		**MaxDuration**	
Feature	Coef	Feature	Coef	Feature	Coef
Node|CPU|Usage	100	*OverallThreshold Check|AvgDuration*	100	*OverallThreshold Check|AvgDuration*	100
Node|CPU1|User	97.53	*CassandraDB| LocalReadCount*	99.02	*Task|GetCommands |ElapsedTimeSum*	99.06
CPU|Usage	97.46	*Disk| FileSystem|writes*	98.88	*DAO|GetResource Metadata| AvgDuration*	98
Node|CPU2|User	97.28	*Network|Transmit Bytes*	98.34	*OverallThreshold Checking|Check Health|AvgDuration*	96
CPU17|User	95.81	*Mem|ActualFree*	98.30	*FeatureRequest| MaxDuration*	96.11
Node|CPU1| Combined	95.28	*Node|Memory| ActualUsed*	98.30	*Task|GetCommands |MaxElapsedTime*	93.58
Node|CPU2|Idle	95.26	*Node|Disk|DBReads*	97.97	*Call|Update ResourceRegion| AvgDuration*	82.21

Table 1: Important Variables for KPIs: Node|PingLatency|Max of Max, Node|PingLatency|Avg of Avg, and OverallThresholdChecking|MaxDuration.

3.2 Reduction of Noise

Class imbalance was a serious problem in the dataset. The considered target KPIs had only 510 positively labeled observations out of 5100 total number of observations. In attempts to achieve better performance, we experimented with several techniques such as undersampling the negative class by different ratios and performing feature selection by various approaches.

Undersampling techniques (removing of the instances in the majority class) has been applied by experimenting with two cases: 50%-positive-vs-50%-negative labels and 35%-positive-vs-65%-negative labels. AUC, Precision and Recall are improving in both cases for all KPIs and algorithms.

Although the results got better from undersampling, they still did not meet our expectations for reasonable analysis using classification models. To improve the results, we have applied several feature ranking/selection methods such as

– Gini Index (it quantifies how often a randomly chosen feature would be incorrectly classified if it were randomly assigned to a data point based on the distribution of the target variable);

– Information Gain (an entropy-based measure evaluating the ability of a feature to discriminate between different classes);

– ReliefF [Konotenko 1994] (it assesses the importance of features based on their ability to distinguish between neighboring instances with the same or different class labels);

– FCBF (Fast Correlation-based Filter) [Yu and Liu 2003]. This technique applies the idea of "predominant correlation". It gives a classifier-independent feature scoring mechanism by selecting features with high correlation with the target variable, but low correlation with other variables. For the correlation, it uses "symmetrical uncertainty" based on information theory and the concepts of entropy and information gain.

After numerous experiments, the results on FCBF ranked dataset outperformed the other models. FCBF ranking was applied after undersampling the negative class in the preceding experiments. When regression task is applied on the datasets with only FCBF non-zero best ranked features, RMSE metric does not substantially decrease, which indicates that both models have high accuracy.

3.3 Results with Undersampling and Feature Ranking

Several classification algorithms such as Decision Tree, CN2 rule induction, Logistic regression, and Naive Bayes were applied to our data set. Previous experiments showed that applying only one method of noise reduction is not enough on this dataset. Thereupon, both undersampling and FCBF ranking (which results in only from 24 to 30 features with non-zero score) were employed to get more predictive models with accurate findings. The dataset with 35% and 65% split (see Table 2 and 3 for two KPIs) performed substantially better, than 50%-vs-50% in terms of evaluation metrics for *OverallThresholdChecking|MaxDuration*, so this dataset will be used for further analysis.

Model	**AUC**	**Precision**	**Recall**
Tree	0.808	0.808	0.806
Naive Bayes	0.838	0.801	0.885
Logistic Regression	0.880	0.806	0.869
CN2 rule inducer	0.826	0.783	0.622

Table 2: Results for 90*-quantile positive labeling for OverallThresholdChecking|MaxDuration. Undersampled* 35%*-vs-*65%*, FCBF ranked.*

However, for *Node|PingLatency|Avg of Avg* there was no noticeable difference between the evaluation metrics for the two datasets, therefore the 35%-vs-65% ratio dataset was used for the experiment (Table 3). As we notice, 35%-vs-65% ratio for positive and negative class labels, respectively, renders the best results. So, further expansion of the positive class does not result in better performance of the models.

Model	AUC	Precision	Recall
Tree	0.684	0.667	0.635
Naive Bayes	0.834	0.765	0.767
Logistic Regression	0.696	0.602	0.759
CN2 rule inducer	0.769	0.728	0.652

Table 3: Results for 90*-quantile positive labeling for node|PingLatency|Avg of Avg. Undersampled* 35%*-vs-*65%*, FCBF ranked.*

3.4 Principal Component Analysis

Large datasets are often difficult to interpret. Principal component analysis (PCA) is a technique for reducing the dimensionality of these datasets, increasing interpretability, and simultaneously minimizing information loss [Jolliffe and Cadima 2016]. We have taken the KPI *OverallThresholdChecking|MaxDuration* with 90-quantile, ran PCA with number of components equal to 380 and got 95% explained variance. So, with only 380 features we can effectively find an optimal representation of the initial data set consisting of 3000 metrics. When comparing results on raw dataset with that of post-PCA, there are noticeable improvements observed. Undersampling the dataset obtained after applying PCA with 35% and 65% ratio and comparing results with the original dataset presented in Table 3, we observed that even the orthogonal components do not appropriately handle the noise. Therefore, there was no need to continue investigations with PCA on the initial data set.

4 Evaluation of Trained Models

To rigorously investigate the issues at KPIs

OverallThresholdChecking|MaxDuration

and

Node|PingLatency|Avg of Avg,

the concepts of variable importance, decision trees, and rule induction are applied. The derived candidate root cause features are compared among the models, intersections are observed, and a list of possible root cause metrics are presented for each KPI. In case of rule induction, when the conditions of the rule are met, i.e., when the features are constrained by the given values, the KPI degrades. So, for a KPI failure instance, the implementation needs to check which of rules are currently satisfied to recommend those for taking actions on.

4.1 Explaining Threshold Checking Duration

The rules corresponding to the positive class with highest Laplace quality obtained by CN2 rule induction are listed in Table 4.

Some rules have common features emphasizing the importance of the variable and its impact on the KPI. The distribution shows the number of observations that comply with the rules with target value equal to 1.

Rule	Quality	Distr
IF *ResourceSymptomRegionUpdate*|*AvgDuration* ≥ 3.234 ms **AND** *SystemAttributes*|*Health* $\geq 91\%$ **THEN** *OverallThresholdChecking*| *MaxDuration* $= 1$	0.938	14
IF *GetUpshards*|*MaxDuration* ≥ 27 ms **AND** *ResourceSymptomRegionUpdate*|*AvgDuration* ≥ 1.4 ms **AND** *SystemAttributes*|*Health* $\leq 78\%$ **AND** *CassandraDB*|*UsedLiveDiskspace* ≤ 892119 *bytes* **AND** *SystemAttributes*|*Health* $\geq 69\%$ **THEN** *OverallThresholdChecking*|*MaxDuration* $= 1$	0.933	13
IF *DAO*|*GetResourceMetadata*|*AvgDuration* ≥ 1 ms **AND** *GetUpshards*|*MaxDuration* ≥ 15 ms **THEN** *OverallThresholdChecking*|*MaxDuration* $= 1$	0.933	13
IF *ResourceSymptomRegionUpdate*|*AvgDuration* ≥ 1.7 ms **AND** *CapacityReclamationSettings*|*MaxDuration* ≤ 9 ms **AND** *CapacityReclamationSettings*|*MaxDuration* ≤ 16 ms **AND** *ControllerDB*|*CPUSystem* ≥ 0.884 KB **THEN** *OverallThresholdChecking*|*MaxDuration* $= 1$	0.929	12
IF *CassandraDB*|*UsedLiveDiskspace* ≥ 984121 *bytes* **AND** *ResourceSymptomRegionUpdate*|*AvgDuration* ≤ 0.776 ms **AND** *SystemAttributes*|*Health* $\geq 75\%$ **AND** *ResourceSymptomRegionUpdate*|*AvgDuration* ≥ 0.721 ms **THEN** *OverallThresholdChecking*|*MaxDuration* $= 1$	0.917	10
IF *CassandraDB*|*UsedLiveDiskspace* ≥ 984121 *bytes* **AND** *CapacityReclamationSettings*|*MaxDuration* ≥ 23 ms **THEN** *OverallThresholdChecking*|*MaxDuration* $= 1$	0.875	6

Table 4: Results for OverallThresholdChecking|MaxDuration.

Overall, the obtained rules and the metrics participating have logical interpretation. The observations show that even in case when the system is healthy, but the average duration of resource symptom region update (a product-specific flow related to evaluating symptoms or unhealthy microstates/fragments at objects under monitoring) has high rate, the abnormality is inevitable. Another interesting rule stresses the importance of the

CassandraDB|*UsedLiveDiskspace*

threshold even when the duration is tolerable.

The decision tree for this KPI is presented in Figure 3.

If the metric

GetUpshards|*MaxDuration*

is greater than 14ms, then 135 out of 168 positively labeled KPI observations can be identified. The features that show duration are also beneficial for root cause analysis, as they reference exactly where the noticeable amount of time was wasted which resulted in the degradation of the KPI.

Another pattern, the metric

ResourceSymptomRegionUpdate|AvgDuraton

found on a branch of the decision tree may indicate that considerable amount of time was wasted on resource's symptom update, which may indicate that the number of symptoms is higher than usual.

The important variables found by Logistic regression and Naive Bayes for this KPI are presented in Table 5.

Besides last two features, the rest were present both in decision tree and CN2 rules, which implies that all four models have outputted the same important variables.

As of kNN regression, the list of important variables obtained from running the model on the raw data has only two matching metrics with this list. When running kNN regression algorithm on the dataset consisting of best ranked features, the number of matching metrics with high coefficients increases.

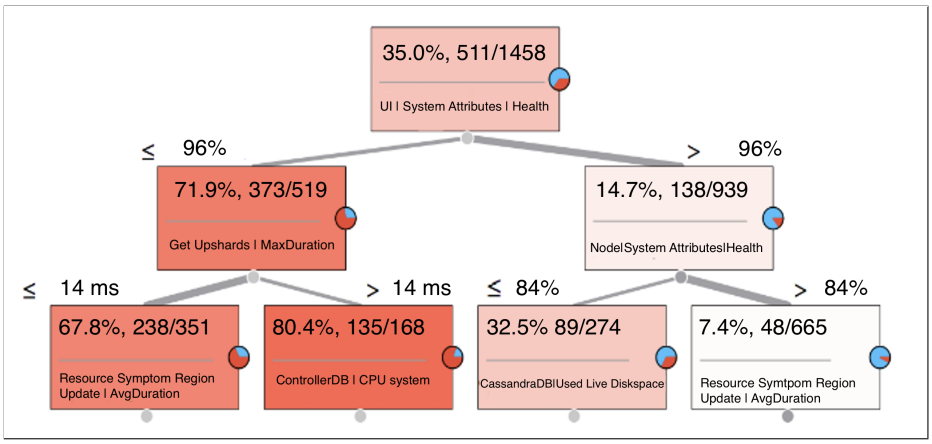

Figure 3: Decision Tree for OverallThresholdChecking|MaxDuration.

CassandraDB|LiveDiskSpaceUsed
Get Upshards|MaxDuration
ResourceSymptomRegionUpdate|AvgDuration
ControllerSQL | CPUSystem
CapacityReclamationSettings | MaxDuration
GemfireClientCalls|LicensingService | ResponsesCount
ReclaimableVmsInfo|Count

Table 5: Important Variable for OverallThresholdChecking|MaxDuration by Logistic regression and Naive Bayes.

4.2 Explaining Node Latency

The rules corresponding to the positive class with highest Laplace quality obtained by CN2 rule induction are listed in Table 6.

It is interesting to notice that the first two rules have more than 30 examples supporting them. Besides high Laplace quality, this is a good measure to validate the accuracy of the results.

From the table it is clear that besides the features showing duration and elapsed time, other system metrics also have their effect on the KPI's behavior.

Metrics such as

$$Network|TransmitBytes$$

and

$$APIService|CurrentHeapSize$$

influence the KPI, as the former is the number of transmit bytes over the network, which naturally affects the latency between the nodes and the latter is the current heap size for API calls. This, interestingly enough, can impact on the latency between the nodes.

Rule	Quality	Distr
IF *Task*|*GetTokens*|*MinimumElapsedTime* ≥ 5 ms **AND** *Call*|*GetSetting*|*AvgDuration* ≥ 3.2 ms **AND** *Service*|*Total number of open file descriptors* ≥ 463 **AND** *Network*|*TransmitBytes* ≥ 61043428 **AND** *API*|*CurrentHeapSize* $\geq$ 201MB **THEN** *Node*|*PingLatency*|*Avg of Avg* $= 1$	0.971	32
IF *API*|*CurrentHeapSize* $\geq$ 409MB **AND** *OverallThresholdChecking*|*MaxDuration* ≥ 855 ms **AND** *Network*|*TransmitBytes* ≤ 115890416 **AND** *NewResourcesCount* ≥ 31 **AND** *Network*|*TransmitBytes* ≥ 71863520 **THEN** *Node*|*PingLatency*|*Avg of Avg* $= 1$	0.947	35
IF *CassandraDB*|*LocalReadCountDelta* ≥ 1803 **THEN** *Node*|*PingLatency*|*Avg of Avg* $= 1$	0.917	10
IF *APICall*|*GetResourceRelationship*|*MinResponseTime* ≥ 73 ms **AND** *APICall*|*GetResourceRelationship*|*MinResponseTime* ≤ 84 ms **AND** *API*|*CurrentHeapSize* $\geq$ 378MB **THEN** *Node*|*PingLatency*|*Avg of Avg* $= 1$	0.938	14
IF *OverallThresholdChecking*|*MaxDuration* ≥ 763 ms **AND** *Call*|*GetSetting*|*AvgDuration* ≥ 13.8 ms **THEN** *Node*|*PingLatency*|*Avg of Avg* $= 1$	0.929	12

Table 6: Rules for Node|pingLatency|Avg of Avg.

The decision tree for this KPI is presented in Figure 4.

The previous KPI

OverallThresholdChecking|MaxDuration

has a significant effect on the current KPI. If the max duration is greater than 755 milliseconds, with 75.4% probability the KPI is anomalous. Moreover, if the count of new resources (objects in the infrastructure) is higher than the number indicated in the tree, 393 out of 503 observations of KPI with positive label are identified. In the case, when the previous KPI is less than or equal to 755 milliseconds, the probability of abnormal behavior of this KPI is relatively small. Metrics like

CassandraDB|LocalReadCountDelta,
Network|TransmitBytes

have logical impact on the KPI whose abnormal behavior is directly affected by them. If the read count is large enough and the transmitted bytes between network exceed the threshold then our KPI has surely abnormal rate.

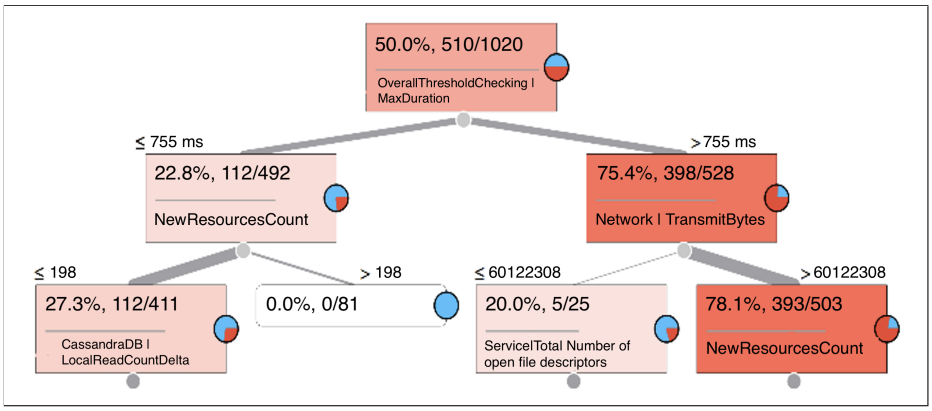

Figure 4: Decision Tree for Node|PingLatency|Avg of Avg.

The important variables found by the Logistic regression and Naive Bayes models for

Node|PingLatency|Avg of Avg

are presented in Table 7.

5 Initial Validation of Results

We have approached the problem of RCA from two perspectives. First, by using regression models we have analyzed the data to get long-term/history-based possible root cause metrics. The space is being narrowed down to some important variables, which impact the KPI's abnormal behavior the most. The kNN regression model applied both on raw and ranked data show that these metrics affect the

OverallThresholdChecking|MaxDuration

degradation the most:

ResourceSymptomRegionupdate|AvgDuration,
CassandraDB|UsedLiveDiskspace,
ControllerDB|CPUsystem,
GemfireClientCalls|LicensingService|RequestsCount.

When applying classification models such as rule induction and decision tree algorithms, incident-based approach is considered. The rules which violation causes KPI degradation are being discovered. So, in case of KPI misbehavior, the implementation can analyze the extracted rules and the metrics that compose those conditions, recognize if the conditions are met and localize the causes with regards to few metrics. The list of potential sources obtained by applying classification models are presented in Table 8. Getting metrics indicating durations of specific actions may seem straightforward, however, this shows exactly what actions consumed most of the time. In addition, getting specific bounds on these metrics show the thresholds which when violated will cause KPI misbehavior. Moreover, it turns out that the proportion of database's used disk space and CPU can negatively affect the amount of time consumed by the system to analyze new observations. All the results are logical and relevant, which are good measures for evaluating root causes.

CassandraDB | LocalReadCountDelta
OverallThresholdChecking | MaxDuration
Network | TransmitBytes
NewResourcesCount
API Service | CurrentHeapSize
Task | GetTaskStatuses | ResponsesRecieved
Task | GetTokens | MinimumElapsedTime
Task | ResourceRegistration | MaxElapsedTime
Call | GetSetting | AvgDuration
Service | Total number of open file descriptors

Table 7: Important Variables for Node|pingLatency|Avg of Avg by Logistic regression and Naive Bayes.

The kNN regression model applied both on raw and ranked data output the following list of important variables affecting the

Node|PingLatency|Avg of Avg

the most:

OverallThresholdChecking|AvgDuration,
Network|TransmitBytes,
FeatureRequest|MaxDuration,
API|CurrentHeapSize.

The list of potential root cause metrics for this KPI obtained from classification models are shown in Table 8.

For this KPI as well, the specific duration metrics are beneficial for troubleshooting the misbehavior of the KPI. An interesting result is the metric

NewResourcesCount

which shows that the count of new resources discovered by the application causes latency increase among the nodes. When particular thresholds for

APIService|CurrentHeapSize

and

CassandraDB|LocalReadCountDelta

are exceeded the misbehavior of the KPI is unavoidable.

The lists mentioned above are the potential sources for the KPIs degradation need to be checked first when that event occurs.

OverallThresholdChecking| MaxDuration	**Node|PingLatency|Avg of Avg**
CapacityReclamationSettings| MaxDuration	*CassandraDB|LocalReadCountDelta*
CassandraDB|LiveDiskSpaceUsed	*OverallThresholdChecking| MaxDuration*
Get Upshards|MaxDuration	*Network|TransmitBytes*
ResourceSymptomRegionUpdate| AvgDuration	*NewResourcesCount*
ControllerDB|CPUSystem	*Task|GetTokens|Minimum ElapsedTime*
	Call|GetSetting|AvgDuration
	Service|Total number of open file descriptors

Table 8: Metric summary for two KPIs (classification scenario).

5.1 Insights Learned

Based on thorough analysis of experimental observations, we get initial insights on the utility of the proposed approach:

- The techniques described in this paper can be used for finding root causes of any KPI degradation if the labeling of data is available or performed in a self-supervised way appropriately.
- The findings discussed above demonstrate that with a relevant dataset of self-metrics from the product deployments at customers, enough powerful models can be trained to predict and explain application/product misbehaviors.

– Tracking the complex conditions constituting the rules we can proactively measure the risk of application failures at the customer's side or in a SaaS delivery of those services. It means that the risk score of the KPI deterioration will increase along with the occurrence of conditions in the rules.

– The methods while productized into cloud management solutions (including their self-healing intelligence engines) can enhance application-aware depth and autonomous capabilities of those data-agnostic services with interpretable recommendations for human administrators.

5.2 Feedback from Product Experts

For rigorous validation of obtained models and rules, as well as importance features, extended studies/surveys on performance troubleshooting of the considered application for a long-term period are required. Such a study should rely on multiple models trained for specific KPI at various environments and quantifying relevant quality measures of recommendations, such as indicative relevance of rules, real importance of derived features in problem resolution, in general, a mean-time-to-repair rate (MTTR). However, setting up this test bed with its comprehensive evaluation over time remains another challenge which might be overcome with a pilot productization of the methods for a set of customers.

In our initial validation of the methodology, we adopted a different approach. The current findings were presented to a small group of experts experienced in troubleshooting this cloud application. The feedback was positive as most of the metrics found were also observed by experts as potential root causes of the given KPI's degradation in customer environments. Some of the metrics found by our analysis were not considered as root causes of the given KPI's anomalous behavior before, however, according to experts there is a logical connection between them, and those metrics should be further evaluated in multiple environments. The extracted rules were of interest to product experts, as they interpreted some of them and confirmed the correctness. Then the goal was to get from the evaluators an overall usefulness rate of discovered patterns. Such a score might attribute a high confidence to our analysis, as the jury consisted of the most experienced product developers and support engineers. This survey demonstrated an approximately 90% utility degree of recommendations presented to them (we need to take into account also that the approximation of real performance issues with generating labels from outlying KPI behaviors might be only close representation of those problems (ground truth)), while 10% remaining patterns were noted with marks on "uncertainty" or "lack of specific knowledge" to be able to verify those recommendations. However, that fraction of patterns was accepted with "surprisingly interesting" mark for further attention in their daily troubleshooting workarounds.

6 Conclusion and Future Work

We demonstrated ways to generate labeled data sets for training interpretable ML models that reveal rules and factors leading to unwanted states of KPIs of cloud applications. The goal of our project is to conduct RCA of KPI degradations with pretrained models while continuously updating it in a separate pipeline. Based on those models we can build troubleshooting and proactive support features (actionable recommender systems)

that automatically alert on potential conditions/rules occurring in the application as an explanation of the KPI misbehavior.

In the modeled use case of application self-diagnostics, those conditions learned from the product usage and self-performance data can be shipped as AI-visible and customer-specific "rules" extracted based on granular time series data. Importance scoring of features is an alternative way to explain long-term behavior of the application and use it for performance optimization as well as situation-based troubleshooting purposes. We also discussed challenges with data labeling using a KPI as a source and techniques to overcome potential noise for training enough accurate ML models. Those models and research results were validated with product experts.

It is worth noting that in our current study we did not focus on the identification of the best/performant classification algorithm or model selection criteria as a primary objective, because of lack of human verified data sets and labeled data on KPI deviations, and, hence, unavailability of benchmarking opportunity. Instead, we discussed the KPI-diagnosis problem from different angles. On the other side, from productization/implementation perspectives, it is more viable and effective that the automated RCA recommender acts on high quality rules obtained using efficient induction algorithms handling noise, such as RIPPER [Cohen 1995]. This is a specific task in our future work plan. In terms of building more reliable RCA recommender systems, such an agenda includes working with other advanced algorithms (XGBoost [Chen and Guestrin 2016]) as well for higher accuracy tree boosting and domain-agnostic explainability frameworks (such as LIME [Ribeira et al 2016]) for local explainability upon availability of human inputs on incident instances on KPIs. Data imbalance might remain an issue in many application scenarios, therefore, alternative ways to overcome this problem need to be considered, e.g. oversampling framework SMOTE [Chawla et al 2011].

We also plan to validate our results further in multiple environments. In addition, it would be interesting to tackle the case when several KPIs need to be explained in various combinations in their behavior. Therefore, multi-labeling and relevant methods/algorithms might be appropriate to research on.

At this stage of our research, highly positive corporate reviews of our approaches (subject to a patented analytics) and models pave the path to productizations in VMware observability platforms ([VMware Aria 2022], [VMware Aria Logs 2023], [VMware Aria Nets 2023], [VMware Tanzu 2022].

Explainable AI is the next phase for many technologies to modernize their solutions for tomorrow's market requirements. We discussed such a challenge in realizing an effective management of large-scale cloud infrastructures and applications in terms of performance diagnosis. Automated RCA is a highly demanded solution in various closely related domains, such as cellular networks [Mdini 2019] and cloud databases [Ma et al 2020] where special and domain-specific modeling are adopted, which are not easily achievable in case of cloud infrastructures, thus leading the research towards more generic and self-supervised approaches.

Acknowledgements

The research is conducted within the ADVANCE Research Grants provided by the Foundation for Armenian Science and Technology. A. Poghosyan and T. Bunarjyan were funded by Science Committee of RA, in the frames of the research project № 20TTAT-AIa014.

References

[Baghdasaryan et al 2022] Baghdasaryan, A., Bunarjyan, T., Poghosyan, A., Harutyunyan, A., El-Zein, J.: On AI-driven customer support in cloud operations, Proc. 3nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2022), Yerevan, Armenia, August 23-24, 32-35 (2022).

[Barredo Arrieta et al 2020] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58, 82-115 (2020).

[Bunarjyan et al 2020] Bunarjyan, T.A., Harutyunyan, A.N., Poghosyan, A. V., Han Vinck, A.J., Chen, Y., Hovhannisyan, N.A.: Estimating efficient sampling rates of metrics for training accurate machine learning models, Proc. 2nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2020), Yerevan, Armenia, September 14-17, 143-152 (2020).

[Chawla et al 2011] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P: SMOTE: Synthetic Minority Over-sampling Technique, https://arxiv.org/abs/1106.1813 (2011).

[Chen et al 2020] Chen, Z, Kang, Y., Li, L., Zhang, X., Zhang, H., Xu, H., Zhou, Y., Yang, L., Sun, J., Xu, Zh., Dang, Y., Gao, F., Zhao, P., Qiao, B., Lin, Q., Zhang, D., Lyu, M.: Towards intelligent incident management: why we need it and how we make it. Proc. 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), 1487–1497 (2020).

[Chen and Guestrin 2016] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system, https://arxiv.org/pdf/1603.02754.pdf (2016).

[Clark and Boswell 1998] Clark, P., Boswell, R.: Rule induction with cn2: Some recent improvements, Proc. European Working Session on Learning (1998).

[Cohen 1995] Cohen, W.: Fast effective rule induction, Proc. Twelfth International Conference on Machine Learning, Tahoe City, California, July 9–12, 115-123 (1995).

[Fürnkranz et al 2012] Fürnkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Springer-Verlag (2012).

[Harutyunyan et al 2014] Harutyunyan, A.N., Poghosyan, A.N., Grigoryan, N.M, Marvasti, M.: Abnormality analysis of streamed log data, Proc. 2014 IEEE Network Operations and Management Symposium (NOMS 2014), Krakow, Poland, May 5-9, 1–7 (2014).

[Harutyunyan et al 2018] Harutyunyan, A.N., Poghosyan, A.V., Grigoryan, N.M., Kushmerick, N., Beybutyan, H.: Identifying changed or sick resources from logs, Proc. IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), Trento, Italy, September 3-7, 86–91 (2018).

[Harutyunyan et al 2019] Harutyunyan, A.N., Poghosyan, A.V., Grigoryan, N.M., Hovhannisyan, N.A., Kushmerick, N.: On machine learning approaches for automated log management, Journal of Universal Computer Science 25(8), 925–945 (2019).

[Harutyunyan et al 2020(1)] Harutyunyan, A.N., Grigoryan, N.M., Poghosyan, A.V., Dua, S, Antonyan, H., Aghajanyan, K., Zhang, B.: Intelligent troubleshooting in data centers with mining evidence of performance problems, Proc. 2nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2020), Yerevan, Armenia, September 14-17, 169-180 (2020).

[Harutyunyan et al 2020(2)] Harutyunyan, A.N., Grigoryan, N.M., Poghosyan, A.V.: Fingerprinting data center problems with association rules, Proc. 2nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2020), Yerevan, Armenia, September 14-17, 159-168 (2020).

[Harutyunyan et al 2022] Harutyunyan, A.N., Aghajanyan, N.K., Harutyunyan, L.A., Poghosyan, A.V., Bunarjyan, T.A., Han Vinck, A.J.: On diagnosing cloud applications with explainable AI, Proc. 3nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2022), Yerevan, Armenia, August 23-24, 23-26 (2022).

[HPE 2022] HPE Systems Insight Manager: www.support.hpe.com/hpesc/public/docDisplay?docId=c05330372 (2022).

[Jolliffe and Cadima 2016] Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments, Philosophical transactions. Series A, Mathematical, physical, and engineering sciences (2016).

[Josefsson 2017] Josefsson, T.: Root-cause analysis through machine learning in the cloud, Master Thesis. Uppsala University: https://uu.diva-portal.org/smash/get/diva2:1178780/FULLTEXT01.pdf (2017).

[Konotenko 1994] Konotenko, I: Estimating attributes: analysis and extensions of RELIEF, Proc. European Conference on Machine Learning, Catania, Italy, April 6-8, 171-182 (1994).

[Lyu and Su 2023] Lyu, M.R., Su, Y.: Intelligent Software Engineering for Reliable Cloud Operations. In: Wang, L., Pattabiraman, K., Di Martino, C., Athreya, A., Bagchi, S. (eds) System Dependability and Analytics. Springer Series in Reliability Engineering (2023).

[Lyu et al 2021] Lyu, Y., Rajbahandur, G.K., Lin, D., Chen, B., Jiang, Z.M., Towards a consistent interpretation of AIOps models, ACM Transactions on Software Engineering and Methodology 31(1), 1–38 (2021).

[Mdini 2019] Mdini, M: Anomaly Detection and root cause diagnosis in cellular networks. PhD Thesis. Rennes (2019).

[Ma et al 2020] Ma, M, Yin, Zh, Zhang, Sh., Wang, Sh., Zeng, Ch., Jiang, X., Hu, H., Luo, Ch.: Diagnosing root causes of intermittent slow queries in cloud databases. PVLDB, 13(8): 1176□1189, 2020.

[Orange 2023] Orange Software: https://orangedatamining.com/.

[Poghosyan et al 2020(1)] Poghosyan, A.V., Harutyunyan, A.N., Grigoryan, N.M., Kushmerick, N.: Learning data center incidents for automated root cause analysis, Proc. 2nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2020), Yerevan, Armenia, September 14-17, 181-190 (2020).

[Poghosyan et al 2020(2)] Poghosyan, A.V., Harutyunyan, A.N., Grigoryan, N.M., Pang, C., Oganesyan, G., Ghazaryan, S., Hovhannisyan, N.: W-TSF: Time series forecasting with deep learning for cloud applications, Proc. 2nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2020), Yerevan, Armenia, September 14-17, 152-158 (2020).

[Poghosyan et al 2021(1)] Poghosyan, A.V., Harutyunyan, A.N., Grigoryan, N.M., Kushmerick, N: Incident management for explainable and automated root cause analysis in cloud data centers, Journal of Universal Computer Science 27(11), 1152-1173 (2021).

[Poghosyan et al 2021(2)] Poghosyan, A.V., Harutyunyan, A.N., Grigoryan, N.M., Pang, C., Oganesyan, G., Ghazaryan, S., Hovhannisyan N.: An enterprise time series forecasting system for cloud applications using transfer learning, Sensors 21(5:1590), 1-28 (2021).

[Poghosyan et al 2022] Poghosyan, A.V., Harutyunyan, A.N., Grigoryan, N.M., Pang, C.: Root cause analysis of application performance degradations via distributed tracing, Proc. 3nd Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2022), Yerevan, Armenia, August 23-24, 27-31 (2022).

[Ribeira et al 2016] Ribeira, M.T., Singh, S., Guestrin, C.: Why should I trust you?: Explaining the predictions of any classifier. arXiv: 1602.04938v1 (2016).

[SDDC 2017] Self-driving cars to self-driving data centers: https://www.broadcom.com/sw-tech-blogs/mainframe/self-driving-cars-self-driving-data-centers (2017).

[Self-Monitoring 2022] Self-Monitoring Metrics for vRealize Operations Manager. https://docs.vmware.com/en/vRealize-Operations/8.6/com.vmware.vcom.metrics.doc/GUID-A286313E-BC21-4965-8850-4A6E4D8E4459 (2022).

[Sole et al 2017] Sole, M., Muntes-Mulero, V., Rana, A.I., and Estrada, G.: Survey on models and techniques for root-cause analysis. arXiv: 1701.08556v2, (2017).

[VMware Aria 2022] VMware Aria Operations: https://www.vmware.com/products/vrealize-operations (2022).

[VMware Aria Logs 2023] VMware Aria Operations for Logs, https://www.vmware.com-/latam/products/vrealize-log-insight (2023).

[VMware Aria Nets 2023] VMware Aria Operations for Networks, https://www.vmware.com-/latam/products/vrealize-network-insight (2023).

[VMware Prods 2023] VMware products, https://www.vmware.com/products (2023).

[VMware Skyline 2022] VMware Skyline: https://www.vmware.com/support/services/skyline (2022).

[VMware Tanzu 2022] VMware Tanzu Observability by Wavefront, https://tanzu.vmware.com-/observability (2022).

[Wang et al 2023] Wang, W., Chen, J., Yang, L., Zhang, H., Wang, Z.: Understanding and predicting incident mitigation time, Information and Software Technology 155 (2023).

[Yu and Liu 2003] Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution, Proc. 20th International Conference on Machine Learning, 2:856–863 (2003).