Towards a Consistent Interpretation of AIOps Models

YINGZHE LYU, Software Analysis and Intelligence Lab (SAIL), Queen's University, Canada
GOPI KRISHNAN RAJBAHADUR (https://orcid.org/0000-0003-1812-5365), DAYI LIN, and BOYUAN CHEN, Centre for Software Excellence, Huawei Canada, Canada
ZHEN MING (JACK) JIANG, Lassonde School of Engineering, York University, Canada

ACM Trans. Softw. Eng. Methodol., Vol. 31, No. 1, Article 16, Publication date: November 2021.
DOI: https://doi.org/10.1145/3488269 (https://doi.org/10.1145/3488269)

Artificial Intelligence for IT Operations (AIOps) has been adopted in organizations in various tasks, including interpreting models to identify indicators of service failures. To avoid misleading practitioners, AIOps model interpretations should be consistent (i.e., different AIOps models on the same task agree with one another on feature importance). However, many AIOps studies violate established practices in the machine learning community when deriving interpretations, such as interpreting models with suboptimal performance, though the impact of such violations on the interpretation consistency has not been studied.

In this article, we investigate the consistency of AIOps model interpretation along three dimensions: internal consistency, external consistency, and time consistency. We conduct a case study on two AIOps tasks: predicting Google cluster job failures and Backblaze hard drive failures. We find that the randomness from learners, hyperparameter tuning, and data sampling should be controlled to generate consistent interpretations. AIOps models with AUCs greater than 0.75 yield more consistent interpretation compared to low-performing models. Finally, AIOps models that are constructed with the Sliding Window or Full History approaches have the most consistent interpretation with the trends presented in the entire datasets. Our study provides valuable guidelines for practitioners to derive consistent AIOps model interpretation.

CCS Concepts: • **Computing methodologies** → **Machine learning**; • **Software and its engineering** → **Software maintenance tools**; **Maintaining software**; **Software evolution**;

Additional Key Words and Phrases: AIOps, model interpretation

ACM Reference Format:
Yingzhe Lyu, Gopi Krishnan Rajbahadur, Dayi Lin, Boyuan Chen, and Zhen Ming (Jack) Jiang. 2021. Towards a Consistent Interpretation of AIOps Models. *ACM Trans. Softw. Eng. Methodol.* 31, 1, Article 16 (November 2021), 38 pages.
https://doi.org/10.1145/3488269 **(https://doi.org/10.1145/3488269)**

1 INTRODUCTION

Ensuring the quality of service of cloud computing platforms is extremely pivotal. A recent IDG survey [32] reports that 92% of organizations leverage cloud computing platforms for running their applications. Therefore, it is extremely important to ensure that these cloud computing platforms remain highly available and efficient, particularly since failures in cloud computing platforms are estimated to cost up to $700 billion annually [58]. For instance, a recent survey [4] pegs the average cost per hour of an organization's server downtime to be anywhere between $300,001 to $400,000.

Cloud computing platforms generate a tremendous amount of data that is impossible to be analyzed manually. Recently, it has become increasingly common for organizations to use **AIOps (Artificial Intelligence for IT Operations)** to leverage such generated data to ensure the quality of service and high availability of cloud computing platforms [6, 8, 21, 42, 62, 88]. AIOps leverages machine learning learners to construct machine learning models (hereafter AIOps models) with operations data collected from the cloud computing platforms (e.g., logs and alert signals) to enable quality assurance tasks such as predicting hard drive failures [42], job termination [22], service outages [88], and performance issues [44]. Note that we use the term “learner” to refer to a machine learning algorithm (e.g., Random Forest) and the term “model” to refer to a trained machine learning model (e.g., a Random Forest model trained on disk failure data).

In addition to using AIOps models to predict failures and outages on a cloud computing platform, several prior studies also interpret AIOps models to identify association between different factors and occurrences of certain failures or outages to make operational and business decisions. For instance, Chen et al. [17] interpret their deep learning-based incident prediction model and find that incident-occurring environment features are one of the most important indicators of a potential incident happening in their platform. Similarly, Zhao et al. [88] interpret their XGBoost-based incident prediction model and find that in their systems, database related issues are the root cause for the incidents. Li et al. [42] interpret their AIOps models to refine their automated alert management system.

These derived interpretations of AIOps models should be consistent (i.e., the feature importance ranking from different models’ interpretations agree with one another) to avoid misleading practitioners. For instance, as we mentioned earlier, Chen et al. [17] derive interpretations from their deep learning-based incident prediction model. However, if their incident prediction model is retrained (e.g., on a local instance of the same training data) and it produces a different set of features being the most important feature associated with incidents, then the practitioners might not know which interpretation to trust. Such a case is indeed possible, as Pham et al. [64] show, when a model is retrained, even on the same training dataset, its performance might change (sometimes even drastically). Thus, it is possible that the interpretations that are derived from the retrained incident prediction model might also vary. More generally, recent studies [75, 76] in several domains point out that many factors (e.g., different hyperparameters) could impact the consistency of derived interpretations of a machine learning model.

However, to the best of our knowledge, none of the prior studies in the field of AIOps investigates the factors that could impact the consistency of the derived interpretations of AIOps models. Though the factors relating to the consistency of derived interpretations have been explored by a handful of studies [36, 66, 75, 76] in other domains, it is pivotal to explore them in the context of AIOps for the following reasons:

- Chen et al. [20] and Dang et al. [21] point out there are several challenges that are unique to the field of AIOps. These unique challenges might present unique factors (which are typically not prevalent in other domains such as machine learning) that impact the consistency of the interpretations derived from AIOps models differently. For instance, as Dang et al. [21] explain, AIOps models are constantly updated/retrained to keep up with the rapidly evolving data. Such rapid retraining of AIOps models could mean that the derived interpretations of an AIOps model could change with every retraining. However, as we explain earlier, consistency of derived interpretation is pivotal for practitioners. Therefore, it is important to understand how the factors that are unique to AIOps impact the consistency of interpretations derived from AIOps.
- The operations data that is typically used to build AIOps models is very different from the data used in other domains like machine learning. Operations data is typically a mixture of temporal (e.g., log data) and spatial (e.g., hardware configurations) data. In addition, it may also contain a mixture of heterogeneous data types such as numeric, ordinal, and nominal values, making it starkly different from data used in other machine learning tasks (which is typically either spatial or temporal in nature).
- As Menzies [57] and Ray et al. [67] show, findings from other domains like machine learning do not necessarily generalize in software analytics domains like AIOps. For example, both Menzies [57] and Ray et al. [67] show that language models used in the machine learning domain typically yield spurious results on software engineering-related data. Furthermore, Menzies [57] warns us that simply using the methods and findings outlined in the field of machine learning on software engineering data might yield suboptimal results.

Therefore, in our study, we aim to better understand the factors that could impact the consistency and the practical adoption of the derived interpretations of an AIOps model. In our article, we first propose a set of rigorous criteria to thoroughly assess the consistency of the interpretations derived from AIOps models. We do so through a case study on two publicly available operations datasets (the Google cluster trace dataset [68] and the Backblaze hard drive statistics dataset [5]) that have been widely used by many prior studies in AIOps [11, 22, 71]. In particular, our proposed criteria investigate how different factors impact the AIOps model interpretation along three key dimensions: **Internal consistency**, **External consistency**, and **Time consistency**.

- **Internal consistency** [64] (a.k.a. model reproducibility) captures the similarity between the derived interpretations of an AIOps model trained from the same setup (i.e., same training data and same implementation) across multiple executions. In other words, internal consistency checks if the interpretations derived from an AIOps model is reproducible. For instance, as Pham et al. [64] show, and a plethora of prior studies [26, 31, 64, 65] warn, unless randomness involved during the training process of machine learning models is controlled, the results might not be reproducible. As a result, the derived interpretations might not be internally consistent, which would make it impossible for the managers and DevOps engineers to choose which interpretation to act on. Despite that, many of the AIOps studies do not take specific steps to ensure reproducibility. For instance, both Zhao et al. [88] and Li et al. [42] use deep learning models to construct their AIOps models. However, neither of those studies take explicit methods to control the randomness involved which could, in turn, impact the reproducibility of the interpretation derived from these AIOps models. Therefore, in RQ1 (Are the interpretations from AIOps models internally consistent?), we study the impact of potential sources of randomness during the model training, which can impact the internal consistency of the derived interpretations of the AIOps models.
  Results. All the three studied sources of randomness (i.e., inherent randomness from learners, randomized hyperparameter tuning, and data sampling randomness) impact the consistency of AIOps model interpretation. Sampling randomness introduces the largest scale of inconsistency to AIOps model interpretation. When all the three sources are controlled during the training of an AIOps model, the derived interpretation of an AIOps model is internally consistent.
- **External consistency** [66] captures the similarity between the derived interpretations of similar-performing AIOps models on a given dataset. In other words, external consistency sanity checks if the models that have the similar performance report similar interpretations for a dataset. Typically, interpretable models are preferred by the practitioners to derive interpretations. Several recent studies in AIOps [42] and machine learning [72] argue that interpretable models even with lower performance are preferred in the AIOps context when high-performing models are not easily interpretable. Rajbahadur et al. [66] recently showed that the derived interpretation between different machine learning models could vary

considerably. In addition, Lipton [49] and Molnar [61] warn that such inconsistency could be exacerbated if low-performing models are used to derive interpretations. Intuitively, interpretations derived from a low-performing interpretable model could be trustworthy only if the interpretable model has the same interpretation as other machine learning models on a given dataset (at least among the similar-performing models). In other words, different models at the same performance level should generate similar interpretations. However, there are no studies that examine the relation between the model performance and model interpretations. In particular, it is not clear if using high-performing models would yield more consistent interpretations than using low-performing models. Hence, in this article, we assess and compare the external consistency of the interpretations from models at different performance levels (i.e., similarities of model interpretations among low-performing models vs. similarities of model interpretations among high-performing models). Such study is even more important for low-performing models, since several prior studies question if low-performing models might generate inconsistent interpretations [37, 49, 61]. Otherwise, interchangeably using models to derive interpretations might be misleading and might result in misguided decisions. Therefore, in RQ2 (Are the interpretations from AIOps models externally consistent?), we investigate the external consistency of derived interpretations of AIOps models at different performance levels.

Results. The interpretations derived from high-performing AIOps models (with a minimum acceptable AUC of 0.75) are more consistent than those from low-performing AIOps models. The interpretations derived from high-performing AIOps models exhibit strong external consistency.

- **Time consistency** [42] captures the similarity between the derived interpretations of an AIOps model across different time periods (i.e., as the AIOps models evolve). In other words, time consistency checks if the interpretation derived from an AIOps model remains generalizable across time. Interpretations derived from AIOps models may be used to make business decisions and process optimizations that have a long-lasting effect. Hence, it is pivotal that the interpretation of an AIOps model should not only just reflect the trend from the most recent data on which it was trained, but also should capture and reflect the trends observed over a longer period of time. However, previous works in defect prediction [7] and AIOps [42] show machine learning models trained on one time period do not generalize well when tested on a different time period and their derived interpretations could also vary depending on the size of the training data. Therefore, in RQ3 (Are the interpretations from AIOps models consistent across time (i.e., time consistent)?), we examine which of the commonly used model updating approaches allows the interpretations of the updated models to be consistent over long periods of time.
  Results. The derived interpretations from AIOps models constructed with the Sliding Window and Full History approaches best capture the trends present across time periods in the entire dataset and exhibit strong time consistency.

The main contributions of our article are the following:

(1) This is the first work that studies the factors that impact the consistency of AIOps model interpretations.

(2) We propose a set of rigorous criteria that enables researchers and practitioners to assess the consistency of their derived interpretations from AIOps models.

(3) We provide several actionable guidelines to improve the consistency of interpretations derived from AIOps models.

(4) To foster replicability of our findings and promote open science, we make the datasets and the code to conduct our study publicly available.

Paper organization. The rest of the article is organized as follows: In Section 2, we present a motivational example that motivates the need for our study in AIOps. In Section 3, we provide the background and related work of AIOps and introduce the research questions that we investigate in our article. In Section 4, we explain our case study setup. Section 5, Section 6, and Section 7 present each of our RQs. Section 8 discusses the results, limitations of the results, and potential future areas for investigation. In Section 9, based on our findings from the results of the studied RQs, we outline several practical guidelines for AIOps researchers and practitioners. In Section 10, we discuss the threats to validity of our study. In Section 11, we conclude our study.

2 A MOTIVATIONAL EXAMPLE

Lei is a DevOps engineer. They are responsible for monitoring and maintaining a batch processing job that runs daily. The batch processing job runs on a cluster and can take hours to finish. In the past, they have noticed that the job may randomly fail and they would need to re-submit the job for execution. Recently the analytics team helped Lei build an AIOps model, which uses the traces collected from the cluster to predict if the job is going to fail in the next half hour. This model is selected from a pool of best-performing candidate models by the analytics team after considering various aspects such as performance and training costs. The analytics team advised Lei to use the AIOps model in two ways:

(1) **Predictive:** When the model predicts the job is going to fail in the next half hour with high confidence, they can manually terminate and re-submit the job to save execution time.

(2) **Explanatory:** By interpreting the trained model, the analytics team advised them that the model learned that one of the most impactful factors that is associated with the job failure is the launch of a specific competing job from another team. Therefore, from an operations perspective they should contact the other team to schedule the competing job at a different time of the day.

Scenario 1: To productionize the model, the analytics team provided Lei with the data and code used to train the model. Lei executed the provided training code on the same data, only to realize that the model they trained provided a different interpretation compared to that of the model trained by the analytics team. In particular, the model they trained does not consider the competing job as an important factor. Lei is confused about which interpretation to trust and whether they should reschedule the competing job with the other team. In this scenario, Lei is concerned about the internal consistency of the AIOps model.

Scenario 2: To avoid performance drift of the model in production, the model needs to be periodically retrained with latest data. However the current model provided by the analytics team is too costly to retrain. Lei asked the analytics team if there is any alternative candidate model that is cheaper to retrain. The analytics team sent over a new, lighter model, which is cheaper to retrain albeit at a slightly reduced performance. However, Lei realized that even though the new model was trained on the same data as the current one, it provides a different interpretation and does not consider the competing job as an important factor, posing a dilemma for Lei about whether to reschedule the competing job with the other team. In this scenario, Lei is concerned about the external consistency of the AIOps model.

Scenario 3: The model was deployed in production and was scheduled to be retrained with new data every day at midnight. Lei contacted the other team about rescheduling the competing job, as advised by the analytics team based on the interpretation of the deployed model. Rescheduling jobs may lead to a lot of downstream administrative and operative changes and therefore is expensive to communicate and perform. However, in just a few days, before the rescheduling was scheduled to take effect, Lei noticed that the recently updated model no longer considers the competing job as an important factor. Such drastic changes in the model interpretation led to a lot of confusion among teams and wasted effort in rescheduling jobs. In this scenario, Lei is concerned about the time consistency of the AIOps model.

3 BACKGROUND

In this section, we first describe the existing work on AIOps. Then, we present the current practices of deriving AIOps interpretation. Finally, we present our criteria for assessing the consistency of AIOps interpretation.

3.1 Existing Work on AIOps

We first introduce the common AIOps applications. Then, we discuss the existing work on AIOps interpretation. Finally, we present the reproducibility concerns in general machine learning tasks.

3.1.1 *AIOps Applications.* Although huge efforts have been devoted to cloud computing systems for ensuring the quality of services, various types of issues (e.g., job termination, hard drive failure, and performance anomalies) are unavoidable. To ensure the reliability of online services, we must resolve and manage these issues in a timely manner, as failing to do so might cause unavailable services and huge financial losses. AIOps solutions contribute to issue management in two phases: (1) AIOps solutions aim to predict whether certain issues would occur by learning from the historical data; and (2) after the issues occur, AIOps solutions aim to help mitigate the issues (e.g., automated problem diagnosis) or provide suggestions to the domain experts (e.g., incident triage).

Issue Prediction. Many of the prior works focus on analyzing monitoring data for predicting the occurrence of various types of issues [11, 19, 22, 29, 40, 41, 42, 44, 47, 56, 71, 82, 88].

Lin et al. [47] and Li et al. [42] predict node failures in large-scale cloud computing platforms by building machine learning models from temporal (e.g., CPU utilization metrics), spatial (e.g., location of a node), and config data (e.g., build data). Similarly, Li et al. [40, 41] build tree-based models to predict hard drive failures. Botezaku et al. [11], Mahdisoltani et al. [56], and Xu et al. [82] leverage SMART-based analysis to build a machine learning pipeline for predicting hard drive failures in large-scale cloud computing platforms. Chen et al. [19] collect and analyze the alert data and its dependencies to predict outages in the whole cloud systems. Zhao et al. [88] propose a deep learning-based approach, *eWarn*, which leverages textual (e.g., keywords in incident tickets) and statistical features (e.g, alert count) to predict incident occurrences. El-Sayed et al. [22] and Rosa et al. [71] predict job failures from trace data collected from Google cloud computing platform. Lim et al. [44] leverage performance metrics to cluster performance issues into the recurrent and unknown ones.

Issue Mitigation. Issues in online services need to be mitigated in a timely manner. Many of the prior works focus on triaging [8, 16, 17], diagnosing [6, 18, 33, 53, 87], and managing issues [35, 48, 50, 51, 83, 84], which benefit the mitigation process.

Triaging. Chen et al. [16] propose a deep learning-based technique to improve the current incident triage process (e.g., distributing the new incident to the responsible team). Chen et al. [17] perform an empirical study on characterizing incidents in online systems and propose DeepIP, a technique to detect incidental incidents (i.e., incidents that are less severe and last for a short period of time), which can reduce the incident triage efforts. Bansal et al. [8] propose DeCaf, a Random Forest-based framework to correlate telemetry data with performance

regressions. In addition, the detected performance regressions are automatically triaged to on-site engineering team.

Diagnosing. Zhang et al. [87] propose an ensemble of models to automatically diagnose performance problems. Chen et al. [18] propose LiDAR, a deep learning-based approach to linking similar incidents based on historical information. Luo et al. [53] mine time-series data and event data to discover correlations between them, which could improve the incident diagnosis process. Banerjee et al. [6] discuss challenges in performance diagnosis in a hybrid-cloud enterprise software environment. Jehangiri et al. [33] present techniques to diagnose performance anomalies using time-series datasets.

Managing. Jiang et al. [35] analyze the similarity between incident descriptions and their corresponding troubleshooting guide to facilitate incident management. Lou et al. [50, 51] develop a software analytic-based system to resolve scalability, reliability, and maintainability of data-driven incident management systems. Xue et al. [83, 84] proactively reduce performance tickets by predicting usage series in cloud data centers. Lin et al. [48] propose a data mining-based technique to detect emerging issues (a sudden burst of new issues) by analyzing historical issues.

3.1.2 *AIOps Interpretation*. The importance of interpretability of machine learning-based systems has drawn increasing attention recently as it is related to the trustworthiness, reliability, and quality of the systems. In the context of AIOps, it is essential that we can reason about the model recommendations to make business decisions (e.g., replacing failing hard drives) and improve the status quo (e.g., improving monitoring infrastructure). Here, we discuss prior studies on interpretations of AIOps solutions.

Li et al. [42] study the importance scores of a random forest-based model and find that alert data is the most important feature set for node failure prediction. Zhao et al. [88] leverage a model-agnostic technique, LIME [69], to generate interpretable report for incident prediction. This technique is mainly used for interpreting each individual prediction. Bansal et al. [8] propose a technique to extract ranked list of rules from random forest-based models, which can be used to explain predicted performance regression. Li et al. [40, 41] present that by using interpretable models (e.g., regression tree-based models), users can derive more meaningful decisions in reducing the hard drive failure rate, which is preferred over ANN-based models in such context. Chen et al. [16] and Jiang et al. [35] all leverage deep learning-based systems for mitigating incidents. The deep learning-based systems contain multiple components for reducing the data noise and adjusting loss functions. They interpret the importance of these components by comparing the performance before and after enabling these services.

3.1.3 *Reproducibility of Machine Learning*. Recent studies [26, 31] have shown that reproducibility has drawn increasing attention in the machine learning field in recent years. Many research papers did not include sufficient information (e.g., dataset and experiment code), which makes it difficult for other researchers to reproduce the experiment results. Furthermore, even with the same experiment setup, there is still variance in the outcomes due to the stochastic nature of machine learning applications [64]. Hence, huge efforts have been devoted to evaluating and improving the reproducibility in the machine learning field. Pham et al. [64] study the variance of deep learning applications and find that non-implementation level factors can cause the accuracy and training time to fluctuate. Pineau et al. [65] propose a reproducibility checklist before submitting papers to NeurIPS to improve the quality of scientific contributions. Various guidelines (e.g., Model Cards [60] and Datasheets [25]) have also been proposed to help practitioners and researchers to improve the reproducibility of their machine learning applications.

The interpretation of machine learning applications also suffers from the reproducibility issues. Fan et al. [23] assess the quality of five interpretation techniques in Android malware analysis applications, where they evaluate the stability, robustness, and effectiveness of the interpretations. Warnecke et al. [81] also study the similar dimensions of interpretation in security domain. They both find that different interpretation techniques could generate different interpretation results for the same prediction result. Other studies [3, 14, 85] also indicate that the interpretation results cannot be trusted when the results cannot be reproduced.

Our study is different from the previous work in three aspects: (1) our study focuses on the consistency of AIOps interpretation—in particular, we study the internal, external, and time consistency by leveraging one interpretation technique; (2) we leverage the model-level interpretation technique, i.e., the feature importance ranking, to extract the general trend and knowledge in the AIOps model, where previous studies mostly focus on instance-level interpretation; and (3) our work is conducted in the AIOps domain with constantly evolving data, which requires the interpretation to be able to capture the general trends across time.

3.2 The Current Practices of Deriving Interpretations from AIOps Models

We first introduce the the overall process of deriving interpretations from AIOps models. Then, we discuss the potential issues with recent AIOps studies.

3.2.1 *The Overall Process*. As we discussed in the previous section, AIOps solutions have been used extensively for a variety of tasks (e.g., incident prediction [42, 88], performance analysis [6, 8], anomaly detection [62], and business decision making [21]). The quality of AIOps solutions is evaluated from various dimensions such as performance, scalability, and maintainability. Although many studies leverage the interpretations of AIOps models to support critical decisions, few studies have been done to systematically assess the quality of interpretations of AIOps models.

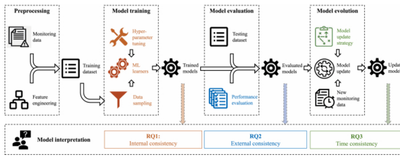

Fig. 1. An overall process of deriving interpretations from AIOps models. Our criteria for assessing the consistency of AIOps model interpretation is highlighted along the process.

Figure 1 shows an overall process of deriving interpretations from AIOps models. We divide the pipeline into four major phases:

(1) The *Preprocessing* phase refers to the process of continuously collecting monitoring data and transforming it into readily available features as inputs for ML models. This is a common practice adopted in many AIOps solutions [42, 55]. In this phase, the training dataset is generated.

(2) The *Model Training* phase refers to the process of training ML models using preprocessed features. It consists of three steps: data sampling, ML learner fitting, and hyperparameter tuning. Data sampling is for selecting a representative subset of a large dataset. ML learner fitting is to select the appropriate learner for the task and fit a model with the learner. Hyperparameter tuning is to find the optimal parameters for the models. In this phase, the trained models are generated.

(3) The *Model Evaluation* phase refers to evaluating the performance of the trained models using the testing dataset. The testing dataset is not seen by the model during the model training phase to prevent data leakage. In this phase, the evaluated models are generated.

(4) The *Model Evolution* phase refers to the practice of deploying the models and constantly updating models. Various model update strategies are applied to reflect the trends contained in the newly available period of monitoring data and mitigate the impact of concept drift [24, 28]. In this phase, the updated models are generated.

The overall process is iteratively conducted until the model performance is above a certain threshold. The interpretation can be derived from the trained models, evaluated models, and the updated models. There are two general approaches to interpret machine learning models: the model-specific approach and the model-agnostic approach. The model-specific approach is mainly used with models that are intrinsically interpretable. Examples of intrinsically interpretable models include linear regression models and decision tree-based models. However, some machine learning models are difficult to interpret due to their complex internal structures (e.g., deep neural networks). To interpret these models, practitioners mainly use the model-agnostic approach that explains the models in a post hoc manner. There are many model-agnostic techniques in the existing literature, such as LIME [69] and permutation feature importance. There are two types of interpretation results that could be produced: model-level interpretation and instance-level interpretation. The model-level interpretation is to understand how the models work internally, while the instance-level interpretation is to explain each single prediction. In this article, we use the permutation feature importance, which is a model-level, model-agnostic interpretation approach, to interpret all the studied AIOps models. We explain the approach and rationale in detail in Section 4.

3.2.2 *Potential Issues with Recent AIOps Studies*.

To understand whether the recent AIOps studies account for the three aforementioned key dimensions along the interpretation consistency, we conducted a literature survey of AIOps studies that derive interpretations from their AIOps models. To survey the literature, we searched Google Scholar with terms including “AIOps,” “Software Engineering,” “Hard Drive Failures,” and “Incident Prediction” to collect the initial set of studies. We then filtered these studies to keep the ones that were published in the past seven years (i.e., after 2014). In the end, 11 studies that we included in our survey were carefully examined by one of the authors. The results are shown in Table 1.

Table 1. The Potential Neglected Key Dimensions of Interpretation Consistency

Paper	Internal Consistency	External Consistency	Time Consistency
P1 [42]	×	×	×
P2 [88]	×		
P3 [8]	×		×
P4 [40]	×		
P5 [41]	×	×	

Paper	Internal Consistency	External Consistency	Time Consistency
P6 [30]	×		×
P7 [86]	×		×
P8 [17]	×		
P9 [35]	×		×
P10 [16]	×		×
P11 [47]	×		×

According to our survey, none of the surveyed papers explicitly mentioned that they control the randomness from the learners, data sampling, and hyperparameter tuning, where applicable. Two out of 11 studies suggested interpreting low-performing models to derive interpretations, while 7 out of 11 studies did not consider or discuss the impact of periodic retraining strategies on the interpretation consistency.

Our summary by no means intends to criticize the prior studies, but to raise awareness that there is currently a gap between recent AIOps studies and the commonly followed practices (e.g., controlling the randomness) in the machine learning community. It is important to take into account if the interpretations derived from AIOps models are internally consistent, externally consistent, or time consistent to avoid misleading decisions from being made by practitioners.

3.3 Our Criteria for Assessing the Consistency of AIOps Interpretation

Here, we describe the motivation of assessing each dimension of AIOps model interpretation consistency in details and formulate the corresponding RQs.

Internal Consistency. Prior work has shown the impact of different random factors (e.g., random sampling of the same dataset and different random seeds for fitting ML learner) in the reproducibility of predictive software engineering [43] and deep learning [64] research, with the focus on model performance reproducibility. As a result, the derived interpretations might not be reproducible (a.k.a. internally consistent). Despite that, many of the AIOps studies did not take actions to ensure reproducibility. To investigate how the randomness from the three steps in the model training phase (highlighted in Figure 1) impacts the internal consistency of interpretations derived from trained models, we formulate the following research question:

> *(RQ1) Are the interpretations from AIOps models internally consistent?*

External Consistency. Many existing studies use interpretable models to compute feature importance ranks without taking into consideration the performance of these interpretable models. The assumption is that these interpretable models are able to capture the general trends in the data despite their lower performance. However, several studies already hint that such a practice could cause inconsistency in the computed insights [49, 61, 66]. Hence, in the model evaluation phase (highlighted in Figure 1), to assess the consistency of the interpretations derived from the evaluated models, we formulate the following research question:

> *(RQ2) Are the interpretations from AIOps models externally consistent?*

Time Consistency. The workloads of large cloud computing platforms are highly dynamic and they tend to evolve significantly throughout its lifetime [42, 47]. To keep up with the constantly evolving, temporal nature of the data, prior studies typically use several model update approaches to keep their AIOps models current and relevant. For instance, some studies periodically retrain their AIOps models on new data [42, 47] or construct time-based ensemble approaches that aggregate local AIOps models trained on small time periods [73, 80]. These models are typically interpreted to make operational decisions [42, 72]. However, the temporal nature of the data might make it hard for the different model update approaches to update the AIOps models to accurately generalize to the underlying trends while still yielding high performance on the most recent data. Hence, to avoid misleading operational decisions, it is important to prevent these updated AIOps models from losing sight of the historical trends in data and overfitting to the most recent time period.

To examine which of the commonly used model updating approaches allows the interpretations of updated models to have the highest time consistency in the model evolution phase (highlighted in Figure 1), we formulate the following research question:

> *(RQ3) Are the interpretations from AIOps models consistent across time (i.e., time consistent)?*

4 CASE STUDY SETUP

In this section, we describe the setup of our case study in detail.

4.1 Studied AIOps Datasets

To understand the challenges of reliably interpreting AIOps models, we perform a case study on two large-scale public AIOps datasets that have been commonly used in prior work [11, 22, 55, 56, 71, 82]: the Google cluster trace dataset [68] (hereinafter referred to as the Google dataset) and the Backblaze hard drive statistics dataset [5] (hereinafter referred to as the Backblaze dataset).

4.1.1 *The Google Dataset*. The Google dataset contains the trace information of job runs on a large-scale cluster at Google. In this study, we use the second version of the dataset[1] that was collected on May 2011, containing 29 days of trace information from a cluster of about 12.5K machines. The dataset includes information about the machines in the cluster, the jobs executed on the cluster, and the tasks under each job. In total, there are 670K jobs and 26M tasks in the dataset. There are four possible states of a job in its lifecycle: unsubmitted, pending, running, and dead. Throughout the lifecycle, a job triggers multiple events that are recorded in the dataset, including submit, schedule, evict, fail, kill, finish, lost, and update.

4.1.2 *The Backblaze Dataset*. The Backblaze dataset contains daily snapshot of statistics of the hard drives in the Backblaze data center. The Backblaze dataset includes hard drive information (e.g., the model and capacity of the hard drive) and **SMART (Self-Monitoring, Analysis and Reporting Technology)** metrics of the hard drives. SMART is a hard drive monitoring system that monitors various indicators of drive reliability to identify imminent hard drive failures. In this study, we use 36 months of Backblaze snapshot data collected from 2015 to 2017, containing 72M records.

4.2 Experiment Context

In this case study, we focus on two AIOps applications: cluster job failure prediction on the Google dataset and hard drive failure prediction on the Backblaze dataset. The interpretation of cluster job failure prediction models can help practitioners identify factors that increase the likelihood of job failure, apply fail-safe mechanisms at the time of job submission, and investigate methods to reduce the occurrence of such risky factors. The interpretation of hard drive failure prediction models can help practitioners understand the early indicators of hard drive failures and improve monitoring infrastructure around the early indicators. Below, we describe the general experiment context of our case study. We describe the RQ-specific experiment designs in the sections of each RQ. A replication package of our experiments and analysis are also provided.[2]

4.2.1 *Data Preprocessing*. Here, we discuss the data preprocessing steps for two datasets separately.

Google cluster job failure prediction: To predict job failures on the Google dataset, we need to first identify whether a job in the dataset finished successfully or not. Such information is not directly provided in the dataset. We use the events of jobs in the dataset as indications for such information. Specifically, we consider a job as failed if its last event in the dataset is a “fail” event. To avoid mislabelling jobs that have not finished running at the time of data collection, we exclude the jobs that started on the last day available in the dataset.

Similar to prior work [22, 55], we extract a set of temporal and configuration metrics as candidate features for training the predictive models. Tantithamthavorn and Hassan [74] stated that highly correlated features should not be used to construct models for interpretation. Therefore, to remove the correlated and redundant features, we use the `varclus` function and the `redun` function in the `Hmisc` R package to identify and remove metrics that show high Spearman correlation and multicollinearity. Table 2 provides a description of metrics used in our study for Google cluster job failure prediction.

We then divide the dataset into subsets with equal time intervals (i.e., periods), based on the date when the job is created. In total, we divide the dataset into 28 one-day periods.

Table 2. Description of Metrics for Google Cluster Job Failure Prediction

Metric	Description
Scheduling class	A number that affects policies for resource access.
Num Tasks	The number of tasks in a job.
Priority	The priority of the job.
Different machine	Whether a task must be scheduled to execute on a different machine than any other currently running task in the job.
Requested CPU	Requested CPU resources.
Requested Disk	Requested disk space resources.

Metric	Description
Mean CPU usage	Mean CPU usage over 5 minutes after job submission.
Mean Memory usage	Mean memory usage over 5 minutes after job submission.
Mean Disk Usage	Mean disk usage over 5 minutes after job submission.
Sd Memory usage	Standard variation of the memory usage over 5 minutes after jobsubmission.

Backblaze hard drive failure prediction: To predict hard drive failures on the Backblaze dataset, we extract the same set of metrics as prior work [55] as candidate features. The dataset is divided into subsets of one-month intervals (i.e., periods) to allow us label the hard drives that fail in the next period and associate them with their metrics in the current period. In total, we obtained 36 one-month periods, with the metrics of each hard drive in the period, and whether each hard drive fails in the next period.

Similar to the Google dataset, we remove metrics that show high Spearman correlation and multicollinearity. Table 3 provides a description of metrics used in our study for Backblaze hard drive failure prediction.

Table 3. Description of Metrics for Backblaze Disk Failure Prediction

Metric	Description
Read Error Rate	Frequency of errors while reading raw data from a disk.
Start/Stop Count^{1}	Number of spindle start/stop cycles.
Reallocated Sectors Count	Quantity of remapped sectors.
Seek Error Rate	Frequency of errors while positioning.
Power-On Hours	Number of hours elapsed in the power-on state.
Power Cycle Count	Number of power-on events.
Reported Uncorrectable Errors	Number of reported uncorrectable errors, the definition is vendor-specific.
Load Cycle Count	Number of cycles into landing zone position.
HDA Temperature	Temperature of a hard disk assembly.
Current Pending Sector Count	Number of unstable sectors (waiting for remapping).
UltraDMC CRC Error Count	Number of CRC errors during UDMA mode.

^1For cumulative SMART attributes, we extract both their raw value from the last day and the difference during the training period as features, following the setup of Lyu et al. [55].

4.2.2 *Model Training*. To ensure the result of our case study is generalizable, we include a variety of learners in our study that have been used in literature [11, 22, 55, 56]: **Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression (LR), Classification And Regression Tree (CART), Gradient Boosting Decision Tree (GBDT), Random Forest (RF)**, and **Multi-layer Perceptron Neural Network (NN)**.

To mitigate the impact of different scales of metrics on model performance and interpretation, we perform data standardization on each metric in the training dataset by removing the mean of the metric and scaling the metric to its unit variance, using the `StandardScaler` function in the `scikit-learn` Python package.

We observe that the dataset is extremely imbalanced, with only 1% failed jobs in the Google dataset and 0.1% failed hard drives in the Backblaze dataset. To mitigate the impact of imbalanced dataset on model performance, we downsample the majority class (i.e., succeed jobs in the Google dataset and normal hard drives in the Backblaze dataset) in the training dataset to a success-to-fail ratio of 10:1 prior to training the model.

The detailed configuration of the model training process is described in sections of each RQ.

4.2.3 *Model Evaluation*. Prior work [74] shows that one should use threshold-independent metrics such as **Area Under the ROC Curve (AUC)** in lieu of threshold-dependent metrics such as Precision, Recall, or F-measure to evaluate model performance. Therefore, in this case study, we use AUC to evaluate the performance of each trained model.

Because of the temporal nature of AIOps datasets, traditional evaluation methods such as cross validation may result in data leakage and therefore lead to inaccurate performance evaluation [55]. In this case study, we evaluate a model that is trained on a specific period of dataset using data in the next period as the testing dataset.

To prepare the testing dataset, we apply the same data scaler that was fitted on the training dataset on the testing dataset. It is worth noting that although we rebalance the training dataset by downsampling the majority class, we do not perform such downsampling on testing dataset.

4.2.4 *Model Interpretation*. There are two types of model interpretation: model-level interpretation (i.e., identifying the features that have the biggest impact on a model's predictions) and instance-level interpretation (i.e., identifying the reason that a model predicts a specific input to be a specific outcome). In this case study, we focus on model-level interpretation, because prior work on AIOps mostly focuses on model-level interpretation [8, 16, 35, 40, 41, 42].

In particular, we use permutation feature importance, a model-agnostic approach to derive the importance of features in a machine learning model in our study. Permutation feature importance evaluates the importance of a feature by randomly shuffling the value of the feature on the testing dataset and measuring the drop of model performance due to such shuffling. We use permutation feature importance in our case study because: (1) some of our studied learners are not intrinsically interpretable; (2) permutation feature importance offers a consistent way to compare across all the models; and (3) many existing studies use permutation feature importance on the interpretations of machine learning models [66, 75]. To control the impact of randomness in permutation feature importance calculation on our case study results, we fix the random seed of the permutation feature importance calculation.

In the rest of the article, we will use the term interpretation and feature importance ranks interchangeably.

4.2.5 *Similarity Measurement of Model Interpretation*. To compare the similarity of interpretations among two or more models, we use the following measurements:

- **Kendall's Tau:** a non-parametric measure of the similarity between two rankings. Kendall's Tau ranges from 0 (no agreement) to 1 (complete agreement). We use the same interpretation schema from prior work [66] to interpret Kendall's Tau in our study:

$$\text{Kendall's Tau Agreement} = \begin{cases} \text{Weak} & \text{if } 0 \leq Tau \leq 0.3. \\ \text{Moderate} & \text{if } 0.3 < Tau \leq 0.6. \\ \text{Strong} & \text{if } 0.6 < Tau \leq 1. \end{cases}$$

- **Kendall's W:** a non-parametric measure of the similarity among multiple rankings. Similar to Kendall's Tau, Kendall's W also ranges from 0 (no agreement) to 1 (complete agreement). We use the same interpretation schema that we use for Kendall's Tau.
- **Top K Overlap Score:** after filtering out features with a negligible importance score (i.e., permutation feature importance score lower than 0.0001), we use the same definition of the Top K Overlap Score as prior work [66]:

$$\text{Top K Overlap Score} = \frac{\cap_{i \geq 2}^{n} \text{Most important K features for model } i}{\cup_{i \geq 2}^{n} \text{Most important K features for model } i},$$

  where n is the number of models for comparison.

5 (RQ1) Are the interpretations from AIOps models internally consistent?

In this section, we evaluate the internal consistency of AIOps model interpretation.

5.1 Approach

Figure 1 shows three components involved in the model training phase of AIOps pipeline: data sampling, hyperparameter tuning, and ML learner. Prior work [43, 64] shows that randomness in the model training phase leads to inconsistent model performance. In this RQ, we hypothesize that randomness in the model training phase also leads to inconsistent model interpretations. In particular, the fitting algorithm of the ML learner may have inherent randomness; the randomized hyperparameter tuning contains randomness; and the random sampling of training dataset also introduces randomness to the model training phase [64].

To validate our hypothesis, we conduct a set of controlled experiments in this RQ: for each of the potential sources of randomness, we control for the other sources of randomness and train the seven studied learners on the two datasets. More specifically:

(1) To evaluate the impact of the inherent randomness from the learner on the internal consistency of AIOps model interpretations, we control for the hyperparameters and the training dataset, i.e., we fix the hyperparameters of each learner to its default hyperparameters and train a model for each studied learner using all data in a given time period and dataset without controlling for the random seed of the

learner's fitting algorithm. The process is repeated 10 times, and the interpretation similarity measurements among the 10 iterations are calculated for each learner on each dataset and its each time period.

(2) To evaluate the impact of the randomness from randomized hyperparameter tuning on the internal consistency of AIOps model interpretations, we control for the inherent randomness from the learner and the training dataset, i.e., for each studied learner, we train a model with randomized search on optimal hyperparameters, using all data in a given time period and dataset, with a fixed random seed for the learner's fitting algorithm. The process is repeated 10 times for each dataset and its each time period, and the interpretation similarity measurements among the 10 iterations are calculated for each learner on each dataset and its each time period.

(3) To evaluate the impact of the randomness from data sampling, we control for the inherent randomness from the learner and the hyperparameters, i.e., for each studied learner, we fix the hyperparameters of the learner to its default hyperparameters, and the random seed for the learner's fitting algorithm, and train a model using a bootstrapped sample in a given time period and dataset. Unlike in Experiments 1, 3, and 4, where we use the same training data (i.e., all data in a given time period) for all iterations to control for training data randomness, here through bootstrapping a different sample of training data in each iteration to intentionally introduce data randomness. The process is repeated 10 times for each dataset and its each time period, and the interpretation similarity measurements among the 10 iterations are calculated for each learner on each dataset and its each time period.

(4) Finally, to evaluate if we can derive an internally consistent interpretation of an AIOps model when controlling for all the above three factors, we conduct a fully controlled experiment: We fix the hyperparameters of each learner to its default hyperparameters and train a model for each studied learner using all data in a given time period and dataset, with a fixed random seed for the learner's fitting algorithm. The process is repeated 10 times, and the interpretation similarity measurements among the 10 iterations are calculated for each learner on each dataset and its each time period.

Table 4 shows an overview of our controlled experiment setup.

Table 4. Controlled Experiment Setup for RQ1

	Potential source of randomness		
Experiment	**ML learner**	**Hyperparameter tuning**	**Data sampling**
1	×	✓	✓
2	✓	×	✓
3	✓	✓	×
4	✓	✓	✓

✓ - Controlled; × - Not Controlled.

5.2 Results

Inherent randomness from learners introduces inconsistencies to the derived interpretations of an AIOps model. Figure 2 shows the interpretation consistency among iterations when controlling for the hyperparameters and the training dataset. As shown in Figure 2, learners such as NN, RF, and CART show inconsistent interpretation among iterations, while LDA, QDA, LR, and GBDT do not show inconsistent interpretation among iterations. It is worth noting that the degree of inherent randomness of a learner's fitting algorithm depends on the specific implementation of such algorithm. While scikit-learn's implementation of LDA and QDA do not involve random states,[3] scikit-learn provides multiple solvers for LR, among which some involve randomness (`sag`, `saga`, or `liblinear`) while others do not.[4] Similarly, XGBoost's implementation of GBDT supports multiple boosters and only shows non-deterministic behavior when a gblinear booster (i.e., Hogwild algorithm) is used.[5] In our study, we used the default solver for LR (i.e., `lbfgs`) and the default booster for GBDT (i.e., `gbtree`), both of which do not involve randomness. Hence, the apparent stability in Figure 2 may be dependent on the implementation of the learners that we used and may not be generalizable to other implementations.

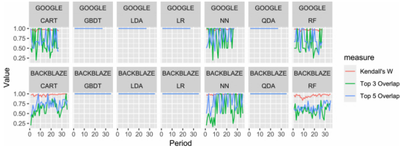

Fig. 2. Consistency of the AIOps model interpretations across iterations under the impact of inherent randomness from learners.

Randomized hyperparameter searching introduces inconsistencies to the interpretation of AIOps models. Figure 3 shows the interpretation consistency among iterations when controlling for the inherent randomness from the learner and the training dataset. As shown in Figure 3, randomized hyperparameter searching introduces inconsistent interpretation among iterations across all studied learners.

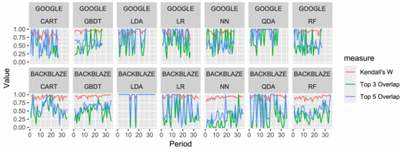

Fig. 3. Consistency of the AIOps model interpretations across iterations under the impact of randomized hyperparameter searching.

Sampling randomness introduces inconsistencies to the interpretation of AIOps models. Figure 4 shows the interpretation consistency among iterations when controlling for the inherent randomness from the learner and the hyperparameters. As shown in Figure 4, sampling (with bootstrap) introduces inconsistent interpretation among iterations across all studied learners.

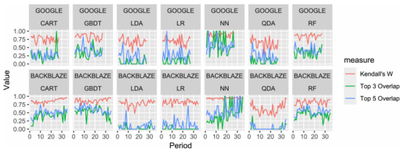

Fig. 4. Consistency of the AIOps model interpretations across iterations under the impact of sampling randomness.

When controlling all three sources of randomness, AIOps model interpretations are internally consistent. Figure 5 shows the interpretation consistency among iterations when controlling for all three sources of randomness. As shown in Figure 5, all learners in every period of both datasets yield identical interpretations across iterations. The result indicates that when the randomnesses of learner, data sampling, and hyperparameter tuning are all controlled, AIOps model interpretation becomes internally consistent.

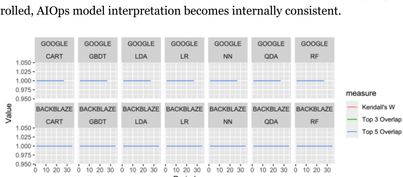

Fig. 5. Consistency of the AIOps model interpretations across iterations when controlling all three sources of randomness.

The randomness from data sampling introduces the largest scale of inconsistency to AIOps model interpretation. Figure 6 provides an aggregated view of the distribution of Kendall's W when different sources of randomness are not controlled, in comparison to a controlled group with all sources of randomness controlled. It can be observed from Figure 6 that generally when randomness from data sampling is introduced, the interpretation of AIOps models show the lowest consistency; followed by hyperparameter tuning, which shows the second-lowest consistency among model interpretations. The ML learner's inherent randomness appears to introduce the least inconsistency to AIOps model interpretation. Such observation is consistent across most learners and datasets, except for **Multi-layer Perceptron Neural Network (NN)**—in the case of NN, hyperparameter tuning introduces comparable interpretation inconsistency to data sampling. A possible explanation is that for NN, the architecture of the network (e.g., the size of hidden layers) are tuned as a group of hyperparameters, which may result in significant difference in the architecture of the trained models. Because of the wide adoption of Neural-Network-based learners in AIOps research (e.g., **Deep Neural Network (DNN)**), future work is needed to further explore the impact of hyperparameter tuning randomness on DNN model interpretation.

Fig. 6. The interpretation consistency (measured by Kendall's W) of models when different sources of randomness are not controlled. Each subplot has four boxplots, from left to right, representing the control group (all sources of randomness controlled), randomness from ML learner, hyperparameter tuning, and data sampling. See Table 4 for detailed setup.

Summary of RQ1

Inherent randomness from learners, randomized hyperparameter searching, and sampling randomness all result in AIOps models yielding different interpretations across different repeated runs. Sampling randomness introduces the largest scale of inconsistency to AIOps model interpretation. By controlling the randomness from the learner, hyperparameter tuning, and data sampling, we are able to derive internally consistent interpretations for AIOps models.

6 (RQ2) Are the interpretations from AIOps models externally consistent?

In this section, we evaluate the external consistency of AIOps model interpretation.

6.1 Approach

Our approach to evaluate the external consistency of AIOps model interpretation consists of three steps: (1) model generation; (2) model clustering; and (3) interpretation comparison.

Step 1: Model generation. In Section 5, we find that the randomness existing in hyperparameter tuning, data sampling, and ML learners themselves would lead to a large variety of models being generated. To preserve the randomness and generate models in different performance scales, we train models on bootstrapped dataset samples with randomized hyperparameter searching. For each of the learners at each period, we repeat the model training process for 10 iterations. During each iteration, a bootstrapped dataset sample is randomly generated without setting explicit random seeds. We apply this process to both studied datasets. In the end, for each dataset and each period, we generated 70 models. For example, the Google dataset contains data of 28 periods, hence, we generated $27 \times 70 = 1,890$ models. We did not use the data from the last period for training, as there would be no available data for testing the performance of models.

Step 2: Model clustering. In this step, we cluster the models trained with the data in same periods based on the model performance (i.e., the AUC metric). We use a widely adopted one-dimensional clustering technique called Jenks natural breaks optimization [34]. To decide the optimal number of clusters, we first conducted the elbow method. The elbow method [78] visualizes the relationship between the number of clusters and the evaluation metric. It is a heuristic commonly used in cluster analysis to decide the optimal number of clusters. The elbow method considers the number of clusters to be optimal if increasing the number of clusters has little impact on the evaluation metrics. In this study, we use **Within Sum of Square (WSS)** as the evaluation metric. WSS is to measure the variability of observations in each cluster and is calculated based on the clustering technique (i.e., Jenks natural breaks optimization in our case). As the number of clusters increases, the WSS will decrease as the variation in each cluster decreases. The optimal cluster number is the cutoff point where increasing cluster number has little impact on WSS.

Figure 7 shows the relations between WSS and the number of clusters for both Google and Backblaze datasets. The x-axis shows the parameters we use in the Jenk natural breaks optimization. For example, *k*_2 represents the final number of cluster is one. From the visualization, we decide that the optimal number of clusters for the Google dataset is 4 (i.e., *k*_5 is the elbow point) and the optimal number of clusters for the Backblaze dataset is 2 (i.e., *k*_3 is the elbow point).

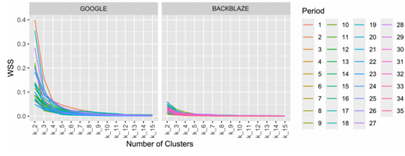

Fig. 7. Elbow graph of performance clustering for deciding the optimal number of clusters.

We then apply Jenks natural breaks optimization to the 70 models in each period of each dataset, with the chosen optimal number of clusters, to group the models into different clusters based on their performance. We noticed that Jenks natural breaks cannot divide the models in Backblaze period 12 into two clusters with at least 2 models in each cluster. We therefore removed Backblaze period 12 from our analysis in the rest of this RQ.

Step 3: Interpretation comparison. In this step, we calculate the interpretation similarity measurements within each of the clusters to check if the interpretations of models in the same performance group are externally consistent. As described in Section 4, we use three types of similarity metrics to measure the consistency of interpretations: the Kendall's W, Top 3 overlap score, and Top 5 overlap score.

To statistically compare the interpretation consistency in different performance groups, we perform the **Wilcoxon Rank Sum (WRS)** test. The Wilcoxon Rank Sum test is an unpaired, non-parametric test commonly used in literature [15, 45, 46], of which the null hypothesis is that for randomly selected values X and Y from two distributions, the probability of X being greater than Y is equal to the probability of Y being greater than X. In this RQ, to confirm if the interpretation consistency of group X is statistically significantly higher than group Y, we use a one-sided alternative hypothesis of X being shifted to the right of Y. Therefore, if the p-value of the Wilcoxon Rank Sum test is less than 0.05, then we conclude that group X has more consistent interpretations within the group than group Y.

To mitigate the threat of false positives during multiple comparisons, we use Bonferroni correction [59] to correct the p-value before concluding a Wilcoxon Rank Sum test where applicable.

The Wilcoxon Rank Sum test only shows whether two distributions are different, but not the magnitude of the difference. To assess the magnitude of the differences, we also calculate the effect size using Cliff's Delta *d*. We use the following schema for interpreting *d*, which is widely used in prior work [15, 45, 46]:

$$\text{Cliff's Delta } d = \begin{cases} \text{Negligible (N)} & \text{if } 0 \leq |d| \leq 0.147. \\ \text{Small (S)} & \text{if } 0.147 < |d| \leq 0.33. \\ \text{Medium (M)} & \text{if } 0.33 < |d| \leq 0.474. \\ \text{Large (L)} & \text{if } 0.474 < |d| \leq 1. \end{cases}$$

6.2 Results

RF and GBDT are the top two best-performing learners for the studied AIOps context. Figure 8 shows the percentage of models trained with each learner in the best-performing clusters. For the Google dataset, Figure 8 shows that RF and GBDT perform much better than other learners. The median percentage of RF and GBDT are around 50%, and the percentage of other five learners are all below 1%. Similarly, the results on the Backblaze dataset show that RF and GBDT are still the top two learners among all seven learners. Different from the results on the Google dataset, the performance of models trained using LDA, QDA, LR, and NN are comparable to those trained using RF and GBDT. In general, none of the other five learners generate more than 25% of models in the best-performing cluster.

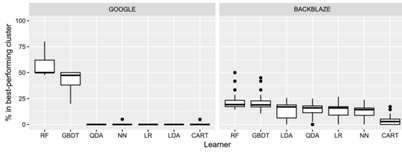

Fig. 8. Percentage of models trained with each learner in the best-performing clusters.

The findings suggest that we should consider using RF and GBDT as the preferred learners in the context of studied AIOps applications and datasets, as they consistently perform the best among all studied learners. In addition, the performance of learners may vary due to different characteristics of different datasets. Future work should further explore the correlation between the AIOps dataset characteristics and performance of different learners.

Models in the highest-performed clusters tend to have more consistent interpretation results compared to lower-performed clusters. Figure 9 shows the three types of similarity measurements for clusters in different performance ranks.

For the Google dataset, the top 1 ranked cluster has higher values in all the three measurements compared to other clusters. For example, the median of Kendall's W in top 1 ranked cluster is 0.81, while the median of Kendall's W in other clusters is only between 0.47 and 0.53. Table 5 presents the detailed results of the comparison. For all three types of measurements, the values of the cluster in rank 1 are statistically higher than the clusters in rank 2,

3, and 4 (corrected p-value <0.05). Furthermore, the results show that all the effect sizes are large. We also observe that the highest-performed cluster in the Google dataset have much more consistent interpretations.

For the Backblaze dataset, there are only two performance clusters: rank 1 and rank 2. We observe that, for the measurements of top 5 overlap score and top 3 overlap score, the differences between rank 1 cluster and rank 2 cluster are significantly different (corrected p-value <0.05). The magnitude of differences are either small or medium. However, for the measurement of Kendall's W, we find that the difference is insignificant. We further investigate the reason for insignificant results of Kendall's W. We notice that the AUCs of the models trained on Backblaze dataset are mostly above 0.75. Based on such observations, we hypothesize that the insignificant difference may be due to the fact that all the trained models have relatively high performance. To verify our hypothesis, we further explore the relationship between performance (i.e., AUC) and within-cluster interpretation similarity measurements (e.g., Kendall's W).

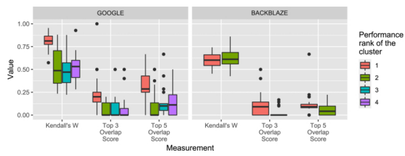

Fig. 9. Distribution of similarity measurements for clusters with different performance ranks.

Table 5. Comparing the Similarity Measurement among Clusters in Different Performance Ranks

Measurement	Dataset	Rank X	Rank Y	Corrected p-value for WRS[+]	Cliff's d*
Kendall's W	Google	1	2	< 0.01	0.71 (L)
		1	3	< 0.01	0.84 (L)
		1	4	< 0.01	0.89 (L)
	Backblaze	1	2	0.91	- (N/A)
Top 5 Overlap Score	Google	1	2	< 0.01	0.73 (L)
		1	3	< 0.01	0.69 (L)
		1	4	< 0.01	0.74 (L)
	Backblaze	1	2	0.01	0.32 (S)
Top 3 Overlap Score	Google	1	2	< 0.01	0.59 (L)
		1	3	< 0.01	0.55 (L)
		1	4	< 0.01	0.60 (L)
	Backblaze	1	2	< 0.01	0.45 (M)

[+] $p < 0.05$ - Significant; $p \geq 0.05$ - Insignificant.

* N/A - Not Applicable; N - Negligible; S - Small; M - Medium; L- Large.

Models that have an AUC of at least 0.75 tend to have high consistency among their interpretation. Figure 10 shows the relationship between a cluster's median AUC value and its within-cluster interpretation consistency represented by Kendall's W value. The blue line is fitted using a Local Polynomial Regression, and the grey area represents a 95% confidence interval. The red line indicates the threshold of Kendall's W (0.6) where the consistency is considered to be high. Note that for the Backblaze dataset, all clusters have a median AUC above 0.70, with the majority of them above 0.75, while for Google dataset, the median AUCs span across 0.5 to 0.95. For the figure of the Google dataset in Figure 10(a), we observe that the within-cluster interpretation similarity begins to increase after AUC is over 0.7 and exceeds a Kendall's W of 0.6 when AUC is larger than 0.75.

From Figure 10(b), we observe that most of the clusters' median AUC values are higher than 0.75, and Kendall's W is consistently above the high agreement threshold. While it is unclear whether the lower AUC values can still result in high Kendalls'W (i.e., > 0.6), from the observable trends in both datasets, we draw a conservative conclusion that models that have an AUC of at least 0.75 tend to have high consistency among their interpretations, and their interpretations are more reliable than models with AUCs less than 0.75.

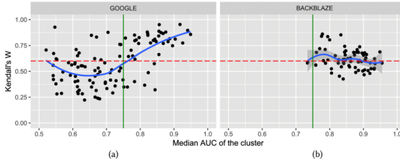

Fig. 10. Relationship between a cluster's median AUC value and its within-cluster interpretation similarity, measured by Kendall's W. The red dashed line marks a Kendall's W of 0.6 (a strong agreement), while the green solid line marks an AUC of 0.75. The blue curve is fitted using a Local Polynomial Regression, and the grey area represents a 95% confidence interval of the fit.

Summary of RQ2

RF and GBDT are the top two best-performing learners for the studied AIOps context. Models in the best performing cluster tend to have more externally consistent interpretations compared to other groups. When the AUC scores are greater than 0.75, the interpretations of AIOps models are externally consistent.

ALGORITHM 1: Streaming Ensemble Algorithm (SEA)

```
no_of_learners_in_ensemble ⟵ 0
E ⟵ ∅ E is the ensemble
time_periods ⟵ time_periods_in_a_given_dataset
n ⟵ number_of_time_periods
ensemble_size ⟵ n/2
while k in time_periods do
    Train learner L_k on k_i
    candidate ⟵ MeanSquaredError(L_k-1, k_i)
    if no_of_learners_in_ensemble ≤ ensemble_size then
        Add L_k-1 to E
    else
        while e in E do
            if candidate < MeanSquaredError(e_j, k_i) then
                replace e_j with candidate
            end if
        end while
    end if
end while
```

7 (RQ3) Are the interpretations from AIOps models consistent across time (i.e., time consistent)?

In this section, we evaluate the time consistency of AIOps model interpretation.

7.1 Approach

To understand how the temporal nature of AIOps data affects the generalizability of the derived interpretation of models updated with different approaches, we first divide the studied datasets into multiple time periods. We divide the Google dataset into 28 one-day time periods and the Backblaze dataset into 36 one-month time periods, as we outline in Section 4.2.1. We then simulate the event of updating the constructed AIOps models to predict the outcome when the last time period becomes available (i.e., the 28th and 36th time periods of Google and Backblaze datasets, respectively). Unlike previous RQs, here, we consider the available data from all the time periods (prior to the last time period) to update the AIOps model. We consider four approaches for updating the AIOps models, including two periodic model retraining approaches (i.e., Sliding Window approach and Full History approach) and two time-based ensemble approaches (i.e., **Accuracy Weighted Ensemble (AWE)** and **Streaming Ensemble Algorithm (SEA)**). Finally, we compare if the derived interpretation of the AIOps models updated with these studied approaches capture the historic trends present in the data through the following three steps: (1) Test model generation, (2) Ground truth extraction, and (3) Interpretation comparison.

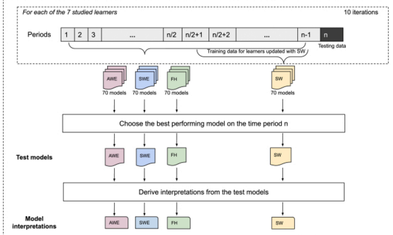

Fig. 11. Overview of Step 1: Test model generation. FH stands for Full History, SEA stands for Streaming Ensemble Algorithm, AWE stands for Accuracy Weighted Ensemble, and SW stands for

Sliding Window.

Step 1: Test model generation. In this step, we first construct the AIOps models (whose interpretation we test) using data points from all the time periods (except the last one) to predict the outcome of data points on the last time period of the studied datasets. Figure 11 presents an overview of our test model generation step. We construct these AIOps models with all the seven learners we outline in Section 4.2.2 and update them using the four model updating approaches. For each learner, we update the associated models with one of the studied model updating approaches, the process is repeated for 10 iterations (with bootstrap sampling), resulting in 70 models being constructed. We do so across both the studied datasets. Among the 70 generated models for each model updating approach, we choose the best-performing model (i.e., the model with the highest AUC on the last time period of each of the studied dataset, a.k.a. test dataset). We then derive the interpretation of the chosen best-performing model (i.e., the test model). At the end, we obtain four derived interpretations from four test models, each from one of the four studied model updating approaches. We describe how each of the studied model updating approaches leverage the historic data and update the model below.

Periodic model retraining approaches. These approaches typically retrain the model whenever data from a new time period becomes available. In our case study, we update the studied AIOps models to predict on the 28th and 36th time periods of Google and Backblaze datasets, respectively.

- *Sliding window approach (SW)*: When updating the AIOps model with an SW approach, we retrain the model with n/2 time periods (i.e., half of the available time periods). For instance, when updating an AIOps model to predict the outcome on 28th time period in Google dataset, we use the last 14 time periods to retrain the model.
- *Full history approach (FH)*: We use all the data available to retrain the AIOps model. For instance, when retraining an AIOps model to predict the outcome on 28th time period in Google dataset, we use all the last 27 time periods.

The time-based ensemble approaches. Instead of updating a single AIOps model, the time-based ensembles combine several AIOps models trained on short periods of time (e.g., models trained on each time period (local models)) in an ensemble [73, 80]. We then use this ensemble model to predict the outcome on future instances. For both the ensemble approaches that we use in this study, we set the ensemble size (i.e., the number of local models that we ensemble together) at n/2, similar to the SW approach. Below, we describe the two ensemble approaches that we consider in our study.

- *Streaming Ensemble Algorithm (SEA)* combines multiple local models using a majority vote. Algorithm 1 presents the working of the SEA. To update the ensemble, SEA replaces the weakest model in the ensemble by observing which of the local models perform the worst in the latest time period for which the SEA is updated. The SEA ensembles multiple learners as follows: When data points from a time period k become available, a new learner L_i (e.g., RF learner) is trained on these data points to build a local model (please see line 7 in Algorithm 1). Then, these data points are used to evaluate the model L_{i-1} trained on the previous time period k_{i-1} (please see line 8 in Algorithm 1). If the number of models in the ensemble has not reached the predetermined ensemble size (n/2 in our case), then the model L_{i-1} is simply appended to the ensemble (please see lines 9–10 in Algorithm 1). Else, the **Mean Squared Error (MSE)** of the model L_{i-1} is compared to all the models in the ensemble. If the MSE of the model L_{i-1} is lower than that of all the models in the ensemble, then the model L_{i-1} replaces the weak learner; if not, the model L_{i-1} is discarded (please see lines 10–16 in Algorithm 1). When the next new set of data points becomes available for training, the model L_i pertaining to the current time period becomes the one that is tested against the other models in the ensemble.
- *Accuracy weighted ensemble (AWE)* is similar to the SEA except on two key points. First, instead of using a majority vote, AWE assigned weights to each model in the ensemble (we detail the weight assignment below). Second, when data from a new time period k becomes available for training, SEA determines if models trained in the previous time period (L_{i-1}) should be kept or removed from the ensemble. Whereas, in AWE, we evaluate if the models (L_i) should be kept or discarded.

The weight for each model in the ensemble is assigned by calculating its prediction error on the latest training data time period k_i. More specifically, we calculate the MSE of each model in the ensemble on the latest training data time period k_i. The MSE for each learner is calculated by computing the difference between the predicted probability of the given model and the observed outcome class (either 0 or 1) in k_i. We note it as MSE_i. The weight for each model in the ensemble is then given by $w_i = MSE_r - MSE_i$, where MSE_r is the MSE value of a model that predicts randomly (i.e., $MSE_r = 0.25$). The models in the ensemble that perform worse than the MSE_r are removed from the ensemble. We use a 10-fold cross-validation to estimate the MSE.

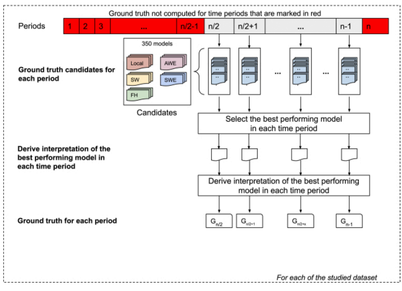

Fig. 12. Overview of Step 2: Ground truth extraction. FH stands for Full History, SEA stands for Streaming Ensemble Algorithm, AWE stands for Accuracy Weighted Ensemble, and SW stands for Sliding Window.

Step 2: Ground truth extraction. In this step, we extract the actual trends (i.e., ground truth) encapsulated in each time period of the studied datasets. Figure 12 presents an overview of our ground truth extraction step. We do so to verify if the derived interpretation of the test models obtained from the previous step are faithful to the historic data. While it is impossible to accurately know what features were the key drivers in prior time periods, as RQ2 results suggest, the interpretation of models in the high-performing clusters are generally consistent. Furthermore, several prior studies also suggest that high-performing models can typically be interpreted reliably [49, 66]. Therefore, to approximate the ground truth of each time period, we pick the best-performing model in that time period and obtain its derived interpretation. For instance, to extract the ground truth of a given time period *n-1*, we pick the model that is trained on *n-1*-th time period (or on all the time periods before it) and has the best performance on the *n*th time period. We then derive the interpretation of this best-performing model as a proxy for the ground truth contained in the time period *n-1*. For example, in Google dataset, if we were to extract the ground truth for time period 25, then we consider the local models trained on 25th time period and models trained on 25th time period that are updated with the studied AIOps model updating approaches. Among these constructed models, if we were to find that the RF model updated with FH approach to be the best-performing model on the 26th time period, we then derive the interpretation of this model. We consider this derived interpretation as the ground truth of the 25th time period.

We compute the ground truth from n/2 to n-1 (where n=#time periods) time periods for each of the studied datasets by extracting the derived interpretation of the best-performing model in the given time period. We do not compute the ground truth on the nthtime period, as we are only trying to observe how faithful the best-performing model on nth time period is to the ground truth available in time periods n/2 to n-1. Furthermore, instead of computing the ground truth on all the time periods starting from 1, we do only from (n/2)th time period, as we include both the AIOps models trained only on one time period (similar to RQ2) and AIOps models trained using the studied model updating approaches. However, the SW approach requires 1 to n/2 time periods to train. For instance, in Google dataset, for a learner to be updated with SW approach, the first 13 time periods are required to update the model trained on 14th time period. We include both the models trained only on one time period and models trained using the studied model updating approaches, since we want to maximize the chance of finding the best-performing model in each time period.

We train all the studied learners that we outline in Section 4.2.2 on time periods n/2 to n-1 of the studied datasets. We first train them using the same approach that we outline in RQ2. On each of the time periods between n/2 and n-1, we first end up with 70 models. Next, we train our studied learners with the four model updating approaches first using time periods 1 to n/2 as the training data to predict on the outcome on (n/2+1)-th time period. We incrementally add each time period from $(n/2+1)^{th}$ until n-1-th time period to the training data. For instance, in the Google dataset, we first build AIOps models on the 14th time period, then we incrementally train models from the 14th to the 27th time period. Therefore, on each time period, we end up with 280 models. Finally, among these $280 + 70 = 350$ ground truth candidate models constructed for each time period between n/2 to n-1, we compute the best-performing model and interpret it. We consider derived interpretation of the best-performing model in each time period as ground truth for the given time period. We do so because it best captures the variation of the dataset during that period.

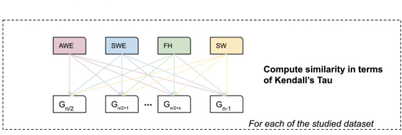

Fig. 13. Overview of Step 3: Test model generation. FH stands for Full History, SEA stands for Streaming Ensemble Algorithm, AWE stands for Accuracy Weighted Ensemble, and SW stands for Sliding Window.

Step 3: Interpretation comparison. To observe if the derived interpretations of the test models are impacted by the temporal nature of the AIOps data, we compare its derived interpretation with the ground truth extracted in each prior time period. Figure 13 presents an overview of our interpretation comparison step. We do so by computing the similarity between the derived interpretation of test models (please note that four test models given by each studied model updating approach) and ground truth of each historic time period in terms of Kendall's Tau. We assert that temporal nature of the AIOps data does not impact the derived interpretation of test models if the

similarity scores given by Kendall's Tau is consistently strong across all historic time periods (i.e., n/2th to n-1-th time periods).

Next, we seek to observe if the derived interpretation of a test model exhibits higher similarity to the ground truth across all the historic time periods compared to the others. If it does, then the model updating approach of the given test model can be thought of as being the most faithful to the trends present in the prior data. To compute that, we conduct a paired Kruskall-Wallis H-test [39] between the computed similarity scores of the test models across all the time periods. A p-value ≤ 0.05 on the Kruskall-Wallis H-test indicates that at least one of test models produces interpretation that has consistently higher similarity with the ground truth across the historic time periods. For instance, best-performing test model updated by SEA to predict the outcome on 28th time period would have 14 similarity scores in the Google dataset. Similarly, the best-performing test model updated by the other studied model updating approaches would also have 14 similarity scores associated with each of them. We would then compute a Kruskall-Wallis H-test between these four similarity score distributions to see if the distribution associated with any one of the studied approaches is higher than the others.

If the Kurskall-Wallis test indicates that at least one of the test model has a different similarity score distribution than others on a given dataset, then we also conduct a pairwise Wilcoxon-rank sum test and Cliff's delta effect size test similar to RQ2. We do so to identify the best model updating approach(es) that yield(s) a significantly superior similarity with the ground truth in comparison to the other studied model updating approaches.

7.2 Results

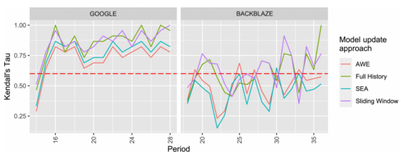

Fig. 14. Similarity measurement (Kendall's Tau) between the interpretation of studied AIOps modelling approaches and estimated ground truth. The red dashed line marks a Kendall's Tau of 0.6 (a strong agreement).

Table 6. Comparing the Difference in Similarity Measurements

		FH		SEA		AWE	
Update Approach	**Dataset**	**Corrected p-value**$^+$	**Cliff's** d^*	**Corrected p-value**$^+$	**Cliff's** d^*	**Corrected p-value**$^+$	**Cliff's** d^*
SW	**Google**	0.86	- (N/A)	< 0.01	0.47 (L)	< 0.01	0.69 (L)
	Backblaze	0.35	- (N/A)	< 0.01	0.49 (L)	0.02	0.64 (L)

$^+$ $p < 0.05$ - Significant; $p \geq 0.05$ - Insignificant. *N/A - Not Applicable; N - Negligible; S - Small; M - Medium; L- Large. (i.e., the similarity between the ground truth interpretation and derived interpretation of an AIOps model updated with the given update approach) between the derived interpretation for AIOps models updated with SW approach and the AIOps models updated with the other studied approaches.

Table 7. Comparing the Difference in Similarity Measurements

		SW		SEA		AWE	
Update Approach	**Dataset**	**Corrected p-value**$^+$	**Cliff's** d^*	**Corrected p-value**$^+$	**Cliff's** d^*	**Corrected p-value**$^+$	**Cliff's** d^*
FH	**Google**	2.19	- (N/A)	0.001	0.47 (L)	0.001	0.63 (L)
	Backblaze	2.68	- (N/A)	0.006	0.57 (L)	0.03	0.40 (L)

$^+$ $p < 0.05$ - Significant; $p \geq 0.05$ - Insignificant. *N/A - Not Applicable; N - Negligible; S - Small; M - Medium; L- Large. (i.e., the similarity between the ground truth interpretation and derived interpretation of an AIOps model updated with the given update approach) between the derived interpretation for AIOps models updated with FH approach and the AIOps models updated with the other studied approaches.

The temporal nature of the AIOps data impact the similarity between the ground truth and the derived interpretation of the AIOps models updated using the studied model updating approaches. From Figure 14, we observe that similarity between the ground truth and derived interpretation of the test models fluctuates across all the time periods for both studied datasets. In particular, except for three instances, none of the AIOps models constructed with the studied approaches have a similarity score of 1 (i.e., a perfect reflection of the ground truth) with the ground truth observed in the studied periods across both studied datasets.

The derived interpretation of AIOps models updated with SW approach and FH approach have consistently higher similarity scores with the ground truth across all historic time periods than models updated with other studied approaches. Figure 14 shows that across the studied time periods, derived interpretation of test models trained with SW and FH approaches have a higher similarity score with the ground truth than the two time-based ensembles. In addition, we also observe that, on Backblaze dataset, test models updated with AWE and SEA approaches have a strong similarity with ground truth only on 4 out of 18 and 1 out of 18 time periods, respectively.

The Kruskall-Wallis test on the computed agreements between the interpretations of the test models and the ground truth resulted in significant p-values ($p < 0.05$) across both datasets. Such a result indicates that at least the similarity scores between the derived interpretation of the test models updated with one the studied model updating approach is significantly different from the similarity measurements produced by the other studied approaches.

Table 6 and Table 7 present the p-values of a pairwise Wilcoxon-rank sum test (p-value corrected) between the similarity scores of test models updated with SW and FH approaches against the computed similarity scores of test models updated with the other studied approaches. From Table 6 and Table 7, we find that the derived interpretation of best-performing test models trained with both the SW model and the FH model produces significantly higher similarity with the ground truth than the test models updated with time-based ensembles approaches across both the studied datasets. Furthermore, we find that the magnitude of effect sizes is consistently large. However, we do not observe significant differences between the similarity measurements of the SW and FH models across either of the studied datasets.

Summary of RQ3

The temporal nature of the AIOps data indeed impacts the similarity between the interpretation of the AIOps models and the ground truth. Among the commonly used approaches to update AIOps models, we find that the derived interpretations from AIOps models updated with SW and FH approaches are the closest to the ground truth across all time periods.

8 DISCUSSIONS AND FUTURE WORK

In this section, we first discuss potential alternative experiment setups that could be applicable to our study and then highlight the gap between AIOps studies and the reproducibility guidelines in the machine learning community. We would like to emphasize that the goal of our study is not to propose an optimal and improved AIOps model, but rather to reveal the threats or pitfalls of prior work's practices when interpreting AIOps models. Therefore, we keep the setup of the study close to prior work. Nevertheless, the alternative experiment setups discussed below can be examined in future work to confirm the generalizability of our findings under different setups.

8.1 Training Setups

In this study, we make several experimental choices in the training setups that we use to train our AIOps models. More specifically,

(1) We choose to standardize the independent metrics before training a model;

(2) We downsample the training dataset prior to training the model;

(3) We do not use any synthetic class rebalancing method like SMOTE.

We do so for two reasons. First, as we mentioned earlier, we wish to keep our AIOps model training setup as close to prior work as possible. All of the aforementioned experimental choices were used in prior work in the field of AIOps. For instance, Lyu et al. [54] standardized the independent metrics of their input data before building their AIOps models. Similarly, several prior studies in AIOps [11, 56] tend to use a downsampling strategy similar to ours.

Second, the experimental choices that we make to build our AIOps models are quite robust. Though we tried to keep the setup of our study as close to prior work as possible, we took care to ensure that the experimental choices that we make are not sub-optimal. As Thomas et al. [77] and Bring [13] state, the derived interpretations of a model are not impacted by the standardization methods such as StandardScaler. Similarly, we do not use advanced class rebalancing methods SMOTE to rebalance our datasets. Several prior studies [74, 75, 79] show that rebalancing the datasets with techniques like SMOTE shifts the distribution of training data and impacts the derived interpretations. Since our study focuses on evaluating the consistency of interpretations derived from AIOps models, we avoided experimental choices that may impact the derived interpretations of a model. However, we highly encourage future research to study the impact of different experimental choices on the consistency of the interpretation derived from AIOps models using the rigorous set of criteria that we outline in our study.

8.2 Evaluation Setups

In this study, we used AUC as the metric to evaluate the performance of models. As stated in Section 4.2.3, we selected AUC as the evaluation metric, because prior work [74] has shown that threshold-independent metrics such as AUC provides more stable evaluation results than threshold-dependent metrics such as precision, recall and F-measure.

In particular, we used **Area Under ROC Curve (AUROC)** in our study. ROC curve, or Receiver Operating Characteristic curve, is a curve that describes the relationship between a model's true positive rate and false positive rate [10]. The AUROC value has a clear interpretation schema. An AUROC value of 0.5 indicates a model performance that is equivalent to random guessing, while an AUROC value of 1 indicates a model that can provide perfect prediction with 0 error rate under a certain threshold. However, when a given dataset is extremely imbalanced (like in many AIOps datasets), AUROC can potentially produce over-optimistic evaluation results.

An alternative metric that is insensitive to imbalanced data is **Area Under Precision-Recall Curve (AUPRC)**, which describes the relationship between a model's precision and recall. However, unlike AUROC, which has a fixed baseline of 0.5, AUPRC does not have a fixed baseline for comparison. In contrast, the baseline to evaluate whether an AUPRC value is good depends on the ratio of positive samples in the dataset [12]. In RQ2, we studied the relationships between model performance and external consistency of the interpretation. To study such relationships, we compared and clustered many different models that are not necessarily trained on the same sample of the dataset. In cases such as those, it is not practical to use AUPRC in our study setup to conduct a fair comparison across models (since AUPRC does not have a fixed baseline for different datasets like AUROC). In addition, using AUPRC would not let us suggest a general acceptable threshold as a guideline for AIOps model interpretation.

8.3 Interpretation Levels

In this study, we used a model-level, model-agnostic approach (i.e., the permutation feature importance) to interpret AIOps models, as prior work on AIOps mostly focus on model-level interpretation. However, in some cases, an instance-level interpretation could be useful for improving the model quality and enhancing the trustworthiness of AIOps models. For example, it is possible that, although two instances have similar features, the prediction results differ (e.g., one predicted as a successful job run and the other predicted as a failed job run). In such cases, it is important to understand the rationale behind the predictions, which could provide insights on how to improve the performance of the AIOps model in the future.

Recently, many instance-level interpretation techniques (e.g., LIME [69], Anchor [70], and SHAP [52]) have been proposed in the research literature [27, 52, 63, 69, 79]. The general idea of instance-level interpretation techniques is to first use local surrogate models (e.g., a linear regression model) to approximate the predictions made by complex models (e.g., a deep neural network model), and then derive interpretations from the local surrogate models that are inherently interpretable. There are two common types of derived interpretations for a prediction: the feature importance ranks and a set of rules and depending on the need both of these can be useful. For instance, Zhao et al. [88] applied LIME to generate a visualized report for AIOps engineers to understand each incident prediction result.

While it is possible (and useful) to apply instance-level interpretation in AIOps models, a recent study [23] in the malware detection domain evaluated five instance-level interpretation techniques. They found that the derived interpretations are not consistent even if the prediction is made by the same model with the same input. Within the five studied instance-level interpretation techniques, LIME [69] achieves the most stable results when the model is slightly changed. Such findings indicate a need for a study similar to ours to provide guidelines to ensure consistent interpretations can be obtained from AIOps models. However, since our study only focused on model-level interpretation, it remains unknown whether our findings apply to instance-level interpretations in general, and we suggest that future work should explore this in detail.

8.4 The Gap between Reproducibility Guidelines in the Machine Learning Community and AIOps Studies

There is a well-accepted reproducibility checklist [65] proposed in the machine learning community that outlines the requirements when submitting research manuscripts to support better reproducibility of the important findings. The checklist contains 21 guidelines that are listed in the appendix of this article. There are five categories of the 21 guidelines: models/algorithms, theoretical claims, datasets, shared code, and experimental results. To understand whether the recent AIOps studies follow the guidelines proposed in the machine learning community, we revisited the 11 previously surveyed AIOps studies in Section 3.2.2.

All the studies that we included in our survey have followed the models/algorithms-related guidelines presented in the reproducibility guideline. None of the surveyed studies propose any theoretical claims, hence guidelines No. 4 and No. 5 do not apply. For the guidelines related to datasets, only 2 out of the 11 studies provided a downloadable link for the dataset that they use in their study. Six out of the 11 studies did not explain the detailed process on how they collect the data if newly presented against guideline No.10. For the guidelines related to shared code, 10 out of 11 studies fail to provide training code, evaluation code, or (pre-) trained models, which goes against guideline Nos. 12, 13, and 14. None of the surveyed studies provide precise commands in the README file to reproduce the results, which goes against guideline No. 15. For the experimental result part, 5 out of 11 studies did not mention the approaches that they use for selecting the hyperparameters. Seven out of 11 studies did not specify the exact number of training and evaluation runs.

Our results show that there is currently a gap between recent AIOps studies and the reproducibility guidelines in the machine learning community. Even though it is understandable that the guidelines are not explicitly followed (for example, for guideline No. 9, it is reasonable that certain datasets cannot be disclosed due to company policies, and the guidelines were only introduced in 2019), it is concerning to find that so many studies violate the guidelines for reproducibility. In addition, even by following the reproducibility guidelines, the results might not be reproducible due to uncontrolled randomness (as we show in RQ1). Hence, our first guideline regarding the internal consistency complements the reproducibility checklist. It is crucial for the future AIOps studies to start accounting for both the reproducibility guidelines and our guidelines to address reproducibility concerns and subsequently achieve the internal consistency of interpretations.

9 GUIDELINES TO PRACTITIONERS

In this section, based on our findings, we recommend the following practical guidelines for AIOps researchers and practitioners to reliably interpret their AIOps models:

[Guideline 1]: ***Always find ways to expose and record the random seeds used to control the non-determinism from the learner, hyperparameter tuning, and data sampling when building an AIOps model.*** From the results presented in Section 5, we observe that identical and internally consistent interpretations could be derived from the constructed AIOps models only when common sources of non-determinism were controlled. It is important to ensure the internal consistency of the derived interpretations of an AIOps model to enable managers and DevOps engineers to act based on these interpretations. For instance, if a DevOps engineer gets a different feature as the source of a problem every time the an AIOps model is retrained (even on the same setup), then it would be impossible to act upon. More importantly, as Krishna and Menzies [38] outline, the DevOps engineer would lose trust in the constructed AIOps model when obtained insights are not consistent.

[Guideline 2]: ***Only use AIOps models with a minimum acceptable performance (i.e., AUC greater than 0.75) to derive interpretations.*** From the results presented in Section 6, we note that derived interpretations of models with low performance (i.e., AUC less than 0.75) exhibit very low external consistency. Only when the model's performance in terms of AUC is greater than 0.75 the derived interpretations among similarly performing models start to become consistent. In light of these findings, we caution against the usage of lower-performing interpretable models in lieu of higher-performing ones for deriving interpretations [42]. Particularly, since interpretations derived from such models may not accurately reflect the ground truth.

[Guideline 3]: ***When using AIOps models to derive interpretations, to update them when new data becomes available, use periodic retraining strategies such as Sliding Window and Full History approaches.*** As we mention in Section 9, it is common for AIOps researchers and practitioners to update their trained AIOps models to keep up with the constant evolution of the AIOps data. When doing so, as we observe from the results presented in Section 7, it is possible for the updated models to lose sight of the historic trends. Only the models updated with Sliding Window and Full History approaches consistently exhibit high similarity with the historic ground truth present in the data.

10 THREATS TO VALIDITY

In this section, we discuss the threats to validity of our case study.

10.1 Internal Validity

In this article, we studied four different AIOps model update approaches and compared their interpretation. The window size of the sliding window approach is set to be a half of the number of total time period, which is consistent with prior work [55].

10.2 External Validity

In this article, we used seven learners on two datasets and leveraged the permutation feature scores for interpretation. The two datasets (the Google cluster trace dataset and the Backblaze hard drive statistics dataset) and the seven learners are widely adopted in previous AIOps studies [42, 55]. Although our results might not generalize when using other learners, datasets, and interpretation methods, our case study setup is generic and can be applied to other types of predictive AIOps tasks in future work.

For AIOps tasks that leverage other types of machine learning algorithms such as unsupervised learning, active learning, or reinforcement learning, the generalizability of our findings should be examined in future work. In addition, our study only focuses on AIOps tasks. Hence, generalizability of our guidelines should be examined in other domains.

In RQ2, we derive a conclusion that models with a minimum AUC value of 0.75 tend to have high consistency among their interpretations. This conclusion is based on the empirical observations made on the two studied datasets. For other datasets and tasks, it is possible that models that do not reach the minimum AUC value threshold (i.e., 0.75) could also have high consistency among their interpretations. Future work is needed to study the generalizability of our conclusions

We implemented our experiments using Python and the 0.24.0 version of scikit-learn API. We did not investigate the impact of different implementations in other programming languages (e.g., R, C++) and other popular machine learning frameworks such as Weka and tensorflow. In addition, we only used the permutation feature importance scores provided by the scikit-learn API to derive the interpretations from each model while there are other implementations [1, 2]. Although we think this technique fits best in our context, there might be other techniques available to quantitatively derive the interpretations. Future study is encouraged to investigate this matter.

10.3 Construct Validity

In the setting of all RQs, we downsample the majority class in the training set so the success-to-fail ratio is 10:1, which may present a threat to our construct validity. However, this process is consistent with prior work to mitigate the imbalance of the dataset [11, 56].

In the setting of RQ1, we repeat the model training process for at least 10 times for each learner to observe the impact of randomness. Similarly, in RQ2, the performance threshold of producing consistent interpretation might be impacted by the characteristics of the randomly sampled datasets. We repeated the training task for each learner in each period of data for at least 10 times to increase the variety of sampled dataset and trained models and mitigate the risk.

To measure the consistency of the interpretations, we used three types of similarity measurements: the Kendall's W, top-five overlap score, and top-three overlap score. These measurements have been widely used in previous studies [55, 66].

We applied an automated process (i.e., randomized search) for hyperparameter tuning in the training process and set the iteration times to be 100. We choose this method over manually tuning, as it would be impractical to manually tune every model with randomly sampled data. In addition, Bergstra and Bengio [9] show that randomized search is efficient in locating the optimal hyperparameters. Even though we cannot guarantee that we extract the best set of hyperparameters, the impact on our study results is limited, since our experiments are conducted on two datasets and seven learners, and our findings do not rely on the hyperparameters being the most optimal.

11 CONCLUSIONS

In this article, we study the consistency of interpretation of AIOps models through a case study on two popular AIOps use cases: (1) job failure prediction using the Google cluster trace dataset; and (2) hard drive failure prediction using the Backblaze hard drive statistic dataset. We assess the consistency of AIOps model interpretation along three dimensions: internal consistency, external consistency, and time consistency. Our results show that inherent randomness from learners, randomized hyperparameter searching, and sampling randomness can lead to internally inconsistent interpretations of AIOps models. In addition, we observe that the performance of AIOps models impact the external consistency of interpretations—the interpretations of AIOps models in the highest-performing clusters tend to be more consistent than those of other models. Finally, we find that AIOps models built using Sliding Window and Full History approaches have the most consistent interpretations to the evolving ground truths.

In light of these findings, we suggest the AIOps researchers and practitioners to: (1) Always control the non-determinism from the learner, hyperparameter tuning, and data sampling by exposing and recording random seeds; (2) Use models with high performance (i.e., AUC greater than 0.75) to derive interpretations; and (3) Use either a Sliding Window or Full History approach to update the model when using the updated model to derive interpretations. Our findings and guidelines can help AIOps researchers and practitioners derive consistent interpretations from AIOps models. For the AIOps applications such as issue prediction and the issue mitigation, by following our guidelines, practitioners can derive more reliable and consistent interpretations that explain the occurrence of failures or outages in the field, which can help them make better managerial decisions that have long-lasting effects and develop better tooling support that saves maintenance efforts.

In the future, we plan to extend our study to more types of AIOps tasks such as performance anomaly detection and diagnosis and other types of interpretation techniques, such as instance-level interpretations. In addition, future work can compare the interpretation between manually tuned models and automated tuned models. We are also interested in applying quality assurance techniques such as metamorphic testing and differential testing to further verify the quality of the model interpretations within the AIOps context.

REFERENCES

[1] Mikhail Korobov, Konstantin Lopuhin. 2021. ELI5. Read The Docs. Retrieved from https://eli5.readthedocs.io/en/latest/overview.html **(https://eli5.readthedocs.io/en/latest/overview.html)**. Navigate to

[2] Ender Celik. 2021. PIMP - R implementation for permutation importance score. RDocumentation. Retrieved from https://www.rdocumentation.org/packages/vita/versions/1.0.0/topics/PIMP **(https://www.rdocumentation.org/packages/vita/versions/1.0.0/topics/PIMP)**. Navigate to

[3] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Proceedings of the Conference on Neural Information Processing Systems*. Navigate to

[4] Thomas Alsop. 2020. Global hourly enterprise server downtime cost 2019. Retrieved from https://www.statista.com/statistics/753938/worldwide-enterprise-serverhourly-downtime-cost/ **(https://www.statista.com/statistics/753938/worldwide-enterprise-serverhourly-downtime-cost/)**. Navigate to

[5] Backblaze. 2020. Hard Drive Data and Stats. Retrieved from https://www.backblaze.com/b2/hard-drive-test-data.html **(https://www.backblaze.com/b2/hard-drive-test-data.html)**. Navigate to

[6] Amitabha Banerjee, Chien-Chia Chen, Chien-Chun Hung, Xiaobo Huang, Yifan Wang, and Razvan Chevesaran. 2020. Challenges and experiences with MLOps for performance diagnostics in hybrid-cloud enterprise software deployments. In *Proceedings of the USENIX Conference on Operational Machine Learning*. Navigate to

[7] Abdul Ali Bangash, Hareem Sahar, Abram Hindle, and Karim Ali. 2020. On the time-based conclusion stability of cross-project defect prediction models. *Empir. Softw. Eng.* 25, 6 (2020), 5047–5083. Navigate to

[8] Chetan Bansal, Sundararajan Renganathan, Ashima Asudani, Olivier Midy, and Mathru Janakiraman. 2020. DeCaf: Diagnosing and triaging performance issues in large-scale cloud services. In *Proceedings of the 42nd International Conference on Software Engineering, Software Engineering in Practice*. Navigate to

[9] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* (2012). Navigate to

[10] Viv Bewick, Liz Cheek, and Jonathan Ball. 2004. Statistics review 13: Receiver operating characteristic curves. *Crit. Care* (2004). Navigate to

[11] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Navigate to

[12] Kendrick Boyd, Vitor Santos Costa, Jesse Davis, and C. David Page. 2012. Unachievable region in precision-recall space and its effect on empirical evaluation. In *Proceedings of the 29th International Conference on Machine Learning*. Navigate to

[13] Johan Bring. 1995. Variable importance by partitioning R 2. *Qual. Quant.* (1995). Navigate to

[14] Oana-Maria Camburu, Eleonora Giunchiglia, Jakob Foerster, Thomas Lukasiewicz, and Phil Blunsom. 2019. Can I trust the explainer? Verifying post-hoc explanatory methods. *CoRR* abs/1910.02065 (2019). Navigate to

[15] Boyuan Chen and Zhen Ming Jack Jiang. 2019. Extracting and studying the logging-code-issue-introducing changes in Java-based large-scale open source software systems. *Empir. Softw. Eng.* (2019). Navigate to

[16] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2019. Continuous incident triage for large-scale online service systems. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*. Navigate to

[17] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2020. How incidental are the incidents? Characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. Navigate to

[18] Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, Zhangwei Xu, and Dongmei Zhang. 2020. Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Navigate to

[19] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, Hao Li, and Yu Kang. 2019. Outage prediction and diagnosis for cloud service systems. In *Proceedings of the World Wide Web Conference*. 2659–2665. Navigate to

[20] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu et al. 2020. Towards intelligent incident management: Why we need it and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1487–1497. Navigate to

[21] Yingnong Dang, Qingwei Lin, and Peng Huang. 2019. AIOps: Real-world challenges and research innovations. In *Proceedings of the 41st International Conference on Software Engineering*. Navigate to

[22] Nosayba El-Sayed, Hongyu Zhu, and Bianca Schroeder. 2017. Learning from failure across multiple clusters: A trace-driven approach to understanding, predicting, and mitigating job terminations. In *Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. Navigate to

[23] Ming Fan, Wenying Wei, Xiaofei Xie, Yang Liu, Xiaohong Guan, and Ting Liu. 2021. Can we trust your explanations? Sanity checks for interpreters in Android malware analysis. *IEEE Trans. Inf. Forens. Secur.* (2021). Navigate to

[24] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. 2004. Learning with drift detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*. Navigate to

[25] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna M. Wallach, Hal Daumé III, and Kate Crawford. 2018. Datasheets for datasets. *CoRR* abs/1803.09010 (2018). Navigate to

[26] Odd Erik Gundersen and Sigbjørn Kjensmo. 2018. State of the art: Reproducibility in artificial intelligence. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*. Navigate to

[27] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. Navigate to

[28] Maayan Harel, Shie Mannor, Ran El-Yaniv, and Koby Crammer. 2014. Concept drift detection through resampling. In *Proceedings of the 31st International Conference on Machine Learning*. Navigate to

[29] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. 2018. Identifying impactful service system problems via log analysis. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Navigate to

[30] Reyhane Askari Hemmat and Abdelhakim Hafid. 2016. SLA violation prediction in cloud computing: A machine learning perspective. *CoRR* arXiv:1611.10338. https://arxiv.org/abs/1611.10338 **(https://arxiv.org/abs/1611.10338)**. Navigate to

[31] Matthew Hutson. 2018. Artificial intelligence faces reproducibility crisis. *Science* (2018). Navigate to

[32] IDG. 2020. 2020 Cloud Computing Study. Retrieved from https://www.idg.com/tools-for-marketers/2020-cloud-computing-study/ **(https://www.idg.com/tools-for-marketers/2020-cloud-computing-study/)**. Navigate to

[33] Ali Imran Jehangiri, Ramin Yahyapour, Philipp Wieder, Edwin Yaqub, and Kuan Lu. 2014. Diagnosing cloud performance anomalies using large time series dataset analysis. In *Proceedings of the 7th International Conference on Cloud Computing*. Navigate to

[34] George Jenks. 1967. The data model concept in statistical mapping. In *International Yearbook of Cartography*. George Philip. Navigate to

[35] Jiajun Jiang, Weihai Lu, Junjie Chen, Qingwei Lin, Pu Zhao, Yu Kang, Hongyu Zhang, Yingfei Xiong, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2020. How to mitigate the incident? An effective troubleshooting guide recommendation technique for online service systems. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Navigate to

[36] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Hoa Khanh Dam, and John Grundy. 2020. An empirical study of model-agnostic techniques for defect prediction models. *IEEE Trans. Softw. Eng.* (2020). Navigate to

[37] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, and Christoph Treude. 2020. The impact of automated feature selection techniques on the interpretation of defect models. *Empir. Softw. Eng.* 25, 5 (2020), 3590–3638. Navigate to

[38] Rahul Krishna and Tim Menzies. 2018. Bellwethers: A baseline method for transfer learning. *IEEE Trans. Softw. Eng.* (2018). Navigate to

[39] William H. Kruskal and W. Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621. Navigate to

[40] Jing Li, Xinpu Ji, Yuhan Jia, Bingpeng Zhu, Gang Wang, Zhongwei Li, and Xiaoguang Liu. 2014. Hard drive failure prediction using classification and regression trees. In *Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Navigate to

[41] Jing Li, Rebecca J. Stones, Gang Wang, Xiaoguang Liu, Zhongwei Li, and Ming Xu. 2017. Hard drive failure prediction using Decision Trees. *Reliab. Eng. Syst. Saf.* (2017). Navigate to

[42] Yangguang Li, Zhen Ming (Jack) Jiang, Heng Li, Ahmed E. Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. 2020. Predicting node failures in an ultra-large-scale cloud computing platform: An AIOps solution. *ACM Trans. Softw. Eng. Methodol.* (2020). Navigate to

[43] Cynthia C. S. Liem and Annibale Panichella. 2020. Run, forest, run? On randomization and reproducibility in predictive software engineering. *arXiv:2012.08387* (2020). Navigate to

[44] Meng-Hui Lim, Jian-Guang Lou, Hongyu Zhang, Qiang Fu, Andrew Beng Jin Teoh, Qingwei Lin, Rui Ding, and Dongmei Zhang. 2014. Identifying recurrent and unknown performance issues. In *Proceedings of the IEEE International Conference on Data Mining*. 320–329. Navigate to

[45] Dayi Lin, Cor-Paul Bezemer, and Ahmed E. Hassan. 2018. An empirical study of early access games on the Steam platform. *Empir. Softw. Eng.* (2018). Navigate to

[46] Dayi Lin, Cor-Paul Bezemer, and Ahmed E. Hassan. 2019. Identifying gameplay videos that exhibit bugs in computer games. *Empir. Softw. Eng.* (2019). Navigate to

[47] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, Murali Chintalapati, and Dongmei Zhang. 2018. Predicting node failure in cloud service systems. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Navigate to

[48] Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, and Dongmei Zhang. 2016. iDice: Problem identification for emerging issues. In *Proceedings of the 38th International Conference on Software Engineering*. Navigate to

[49] Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.*Queue* (2018). Navigate to

[50] Jian-Guang Lou, Qingwei Lin, Rui Ding, Qiang Fu, Dongmei Zhang, and Tao Xie. 2013. Software analytics for incident management of online services: An experience report. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. Navigate to

[51] Jian-Guang Lou, Qingwei Lin, Rui Ding, Qiang Fu, Dongmei Zhang, and Tao Xie. 2017. Experience report on applying software analytics in incident management of online service. *Autom. Softw. Eng.* (2017). Navigate to

[52] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Navigate to

[53] Chen Luo, Jian-Guang Lou, Qingwei Lin, Qiang Fu, Rui Ding, Dongmei Zhang, and Zhe Wang. 2014. Correlating events with time series for incident diagnosis. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Navigate to

[54] Yingzhe Lyu. 2021. Replication Package for an Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions. Retrieved from **https://github.com/SAILResearch/suppmaterial-19-yingzhe-aiops_data_splitting (https://github.com/SAILResearch/suppmaterial-19-yingzhe-aiops_data_splitting)**. Navigate to

[55] Yingzhe Lyu, Heng Li, Mohammed Sayagh, Zhenming (Jack) Jiang, and Ahmed E. Hassan. 2021. An empirical study of the impact of data splitting decisions on the performance of AIOps solutions. *ACM Trans. Softw. Eng. Methodol.* (2021). Navigate to

[56] Farzaneh Mahdisoltani, Ioan A. Stefanovici, and Bianca Schroeder. 2017. Proactive error prediction to improve storage system reliability. In *Proceedings of the USENIX Annual Technical Conference*. 391–402. Navigate to

[57] Tim Menzies. 2019. The five laws of SE for AI. *IEEE Softw.* 37, 1 (2019), 81–85. Navigate to

[58] Kit Merker. 2020. Council Post: How to Optimize Gross Margins For Digital Services. Retrieved from **https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2020/12/03/how-to-optimize-gross-margins-for-digital-services/?sh=13d62bb130c3 (https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2020/12/03/how-to-optimize-gross-margins-for-digital-services/?sh=13d62bb130c3)**. Navigate to

[59] Rupert G. Miller. 2012. *Simultaneous Statistical Inference*. Springer Science & Business Media. Navigate to

[60] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. Navigate to

[61] Christoph Molnar. 2019. *Interpretable Machine Learning*. Retrieved from **https://christophm.github.io/interpretable-ml-book/ (https://christophm.github.io/interpretable-ml-book/)**. Navigate to

[62] Sasho Nedelkoski, Jorge S. Cardoso, and Odej Kao. 2019. Anomaly detection and classification using distributed tracing and deep learning. In *Proceedings of the 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. Navigate to

[63] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. 2019. Meaningful explanations of Black Box AI decision systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Navigate to

[64] Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yaoliang Yu, and Nachiappan Nagappan. 2020. Problems and opportunities in training deep learning software systems: An analysis of variance. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Navigate to

[65] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. 2020. Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program). arxiv:2003.12206 [cs.LG] Navigate to

[66] Gopi Krishnan Rajbahadur, Shaowei Wang, Gustavo A. Oliva, Yasutaka Kamei, and Ahmed E. Hassan. 2021. The impact of feature importance methods on the interpretation of defect classifiers. In *IEEE Trans. Softw. Eng.* Navigate to

[67] Baishakhi Ray, Vincent Hellendoorn, Saheel Godhane, Zhaopeng Tu, Alberto Bacchelli, and Premkumar Devanbu. 2016. On the "naturalness" of buggy code. In *Proceedings of the IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 428–439. Navigate to

[68] John Wilkes. 2020. *Google Cluster-Usage Traces V3*. Technical Report. Google Inc. Posted at **https://github.com/google/cluster-data/blob/master/ClusterData2019.md (https://github.com/google/cluster-data/blob/master/ClusterData2019.md)**. Navigate to

[69] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Navigate to

[70] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Navigate to

[71] Andrea Rosà, Lydia Y. Chen, and Walter Binder. 2015. Catching failures of failures at big-data clusters: A two-level neural network approach. In *Proceedings of the 23rd IEEE International Symposium on Quality of Service*. Navigate to

[72] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* (2019). Navigate to

[73] W. Nick Street and YongSeog Kim. 2001. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Navigate to

[74] Chakkrit Tantithamthavorn and Ahmed E. Hassan. 2018. An experience report on defect modelling in practice: Pitfalls and challenges. In *Proceedings of the International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP'18)*. Navigate to

[75] Chakkrit Tantithamthavorn, Ahmed E. Hassan, and Kenichi Matsumoto. 2018. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. *IEEE Trans. Softw. Eng.* 46, 11 (2018), 1200–1219. Navigate to

[76] Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E. Hassan, and Kenichi Matsumoto. 2018. The impact of automated parameter optimization on defect prediction models. *IEEE Trans. Softw. Eng.* 45, 7 (2018), 683–711. Navigate to

[77] D. Roland Thomas, Edward Hughes, and Bruno D. Zumbo. 1998. On variable importance in linear regression. *Soc. Indic. Res.* (1998). Navigate to

[78] Robert L. Thorndike. 1953. Who belongs in the family. *Psychometrika* (1953). Navigate to

[79] Burak Turhan. 2012. On the dataset shift problem in software engineering prediction models. *Empir. Softw. Eng.* (2012). Navigate to

[80] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Navigate to

[81] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. 2020. Evaluating explanation methods for deep learning in security. In *Proceedings of the IEEE European Symposium on Security and Privacy*. Navigate to

[82] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, Murali Chintalapati, and Dongmei Zhang. 2018. Improving service availability of cloud systems by predicting disk error. In *Proceedings of the USENIX Annual Technical Conference*. Navigate to

[83] Ji Xue, Robert Birke, Lydia Y. Chen, and Evgenia Smirni. 2016. Managing data center tickets: Prediction and active sizing. In *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Navigate to

[84] Ji Xue, Robert Birke, Lydia Y. Chen, and Evgenia Smirni. 2018. Spatial-temporal prediction models for active ticket managing in data centers. *IEEE Trans. Netw. Serv. Manag.* (2018). Navigate to

[85] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I. Inouye, and Pradeep Ravikumar. 2019. On the (In)fidelity and sensitivity of explanations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. Navigate to

[86] Xuezhi Zeng, Saurabh Garg, Mutaz Barika, Sanat Kumar Bista, Deepak Puthal, Albert Y. Zomaya, and Rajiv Ranjan. 2021. Detection of SLA violation for big data analytics applications in cloud. *IEEE Trans. Comput.* (2021). Navigate to

[87] Steve Zhang, Ira Cohen, Moisés Goldszmidt, Julie Symons, and Armando Fox. 2005. Ensembles of models for automated diagnosis of system performance problems. In *Proceedings of the International Conference on Dependable Systems and Networks*. Navigate to

[88] Nengwen Zhao, Junjie Chen, Zhou Wang, Xiao Peng, Gang Wang, Yong Wu, Fang Zhou, Zhen Feng, Xiaohui Nie, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2020. Real-time incident prediction for online service systems. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Navigate to

Footnotes

[1] https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md **(https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md)**.

[2] https://github.com/YingzheLyu/AIOpsInterpretation **(https://github.com/YingzheLyu/AIOpsInterpretation)**. The replication package is set to be private during the review period and will be made public once the article is accepted. Please use the following GitHub account to view the repository in the meanwhile. Username: `reviewer-AIOpsInterpretation`, Password: `gUU!9sL9UDPT`

[3] https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/discriminant_analysis.py **(https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/discriminant_analysis.py)**.

[4] https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/linear_model/_logistic.py **(https://github.com/scikit-learn/scikit-learn/blob/15a949460/sklearn/linear_model/_logistic.py)**.

[5] https://xgboost.readthedocs.io/en/latest/python/python_api.html **(https://xgboost.readthedocs.io/en/latest/python/python_api.html)**.

Yingzhe Lyu, Gopi Krishnan Rajbahadur, Dayi Lin, and Boyuan Chen contributed equally to the work.

Authors' addresses: Y. Lyu, Software Analysis and Intelligence Lab (SAIL), Queen's university, 133 Princess St, Kingston, Kingston, Ontario, Canada, K7L 1A8; email: ylyu@cs.queensu.ca **(mailto:ylyu@cs.queensu.ca)**; G. K. Rajbahadur, D. Lin, and B. Chen, Dayi Lin, Boyuan Chen – Centre for Software Excellence, Huawei, 275 Queen Street, Kingston, Ontario, Canada K7K 3T1; emails: gopi.krishnan.rajbahadur1@huawei.com **(mailto:gopi.krishnan.rajbahadur1@huawei.com)**, dayi.lin@huawei.com **(mailto:dayi.lin@huawei.com)**, boyuan.chen1@huawei.com **(mailto:boyuan.chen1@huawei.com)**; Z. M. (Jack) Jiang, Department of Electrical Engineering & Computer Science, York University, Toronto, Ontario, Canada M3J 1P3; email: zmjiang@cse.yorku.ca **(mailto:zmjiang@cse.yorku.ca)**.

1049-331X/2021/11-ART16 $15.00
DOI: https://doi.org/10.1145/3488269 **(https://doi.org/10.1145/3488269)**

Publication History: Received March 2021; revised June 2021; accepted August 2021