Noname manuscript No.
(will be inserted by the editor)

Can We Recycle Our Old Models? An Empirical Evaluation of Model Selection Mechanisms for AIOps Solutions

Yingzhe Lyu · Hao Li · Heng Li · Ahmed E. Hassan

Received: date / Accepted: date

Abstract AIOps (Artificial Intelligence for IT Operations) solutions leverage the tremendous amount of data produced during the operation of large-scale systems and machine learning models to assist software practitioners in their system operations. Existing AIOps solutions usually maintain AIOps models against concept drift through periodical retraining, despite leaving a pile of discarded historical models that may perform well on specific future data. Other prior works propose dynamically selecting models for prediction tasks from a set of candidate models to optimize the model performance. However, there is no prior work in the AIOps area that assesses the use of model selection mechanisms on historical models to improve model performance or robustness. To fill the gap, we evaluate several model selection mechanisms by assessing their capabilities in selecting the optimal AIOps models that were built in the past to make predictions for the target data. We performed a case study on three large-scale public operation datasets: two trace datasets from the cloud computing platforms of Google and Alibaba, and one disk stats dataset from the BackBlaze cloud storage data center. We observe that the model selection mechnisms utilizing temporal adjacency tend to have a better performance and can prevail the periodical retraining approach. Our findings also highlight a performance gap between existing model selection mechnisms and the theoretical upper bound which may motivate future researchers and practitioners in investigating more efficient and effective model selection mechanisms that fit in the context of AIOps.

Yingzhe Lyu, Hao Li, Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University, Kingston, Ontario, Canada
E-mail: {ylyu, ahmed}@cs.queensu.ca, hao.li@queensu.ca

Heng Li
Department of Computer and Software Engineering
Polytechnique Montreal, Montreal, Quebec, Canada
E-mail: heng.li@polymtl.ca

1 Introduction

Modern large-scale software systems can generate tremendous amounts of data during their daily operations. As the volume of operation data grows, it is getting increasingly challenging for practitioners to manually analyze and utilize such data. AIOps (Artificial Intelligence for IT Operations), which helps practitioners automatically leverage such rich information on the system's runtime behavior and health condition [49], has gained increasing popularity among practitioners and researchers. Today, AIOps solutions support various goals in software and system operations, such as machine failure predictions [33,30], job failure predictions [20,51], disk failure predictions [65,8,20], performance degradation detection [9,31], and service outage predictions [14]. Many proposed AIOps solutions have already demonstrated promising benefits in practice [30,27,4,62,5]. For example, Lin et al. [33] successfully applied their technique on one of Microsoft's large-scale cloud service systems to predict potential node failures based on historical data. Botezaku et al. [8] leverage monitoring information to build a machine learning pipeline for predicting hard drive failures on large-scale cloud computing platforms.

Despite advances in ML models and their applications in AIOps, many challenges are still associated with the maintenance and data evolution of AIOps solutions following their deployment in the field. Concept drift [60,59, 58,45], which refers to the change in data distribution and in the relationship between the variables, can lead to the obsolescence of models trained on stale data and negatively impact the performance in the future time [30]. Our prior works [40,39] find that the operation datasets are subject to a considerable scale of concept drift, negatively affecting the model performance and stability in the field. In order to mitigate the issue of concept drift, practitioners should conduct constant maintenance and update to sustain the model performance and stability once deployed in the field [40,39,15]. However, existing AIOps studies either train a static model regardless of the potential threat from concept drift [20,51,8,42,14,67] or periodically retrain the model to maintain performance and stability [30,15,33]. Stationary models may suffer from performance degradation and impact user experience in the field, while periodically retraining the model could be very expensive (e.g., resources and human efforts involved in updating, verifying, and integrating the model) [30].

As reported in previous work, ensuring the quality of software systems is still an open challenge for the research community [46], and we set out to find more efficient tools to maintain AIOps solutions. Previous works [16,73] suggest that model decision-making mechanisms can benefit model performance and robustness when faced with concept drift, as they leverage the diversity of models to provide robust predictions in dynamic and changing conditions. As AIOps models deployed in the field require constant retraining to maintain their performance and steer away from obsolescence [27,33,30,65], a large number of historical models would accumulate during this process. Some useful information will likely still be buried in these historical models and can be salvaged and recycled to improve prediction performance and overall performance stability. However, there is no prior work and little empirical evidence in the AIOps area that assesses the use of model selection mechanisms in improving model performance and robustness. To fill the gap, we propose to examine the application of model selection mechanisms on the historical models for improving model performance and robustness while introducing minimal cost, namely *model selection mechanisms on historical models*. We conduct a case study to evaluate various forms of selection mechanisms on historical models in AIOps solutions that select the best models in history to make predictions, and assess these model selection mechanisms' performance and robustness on operation datasets. This study aims to address the following research questions:

– RQ1: How well can the model selection mechanisms achieve optimal performance for AIOps solutions?
– RQ2: How well can the model selection mechanisms achieve optimal model ranking for AIOps solutions?
– RQ3: How stable are the model ranking results achieved by the model selection mechanisms?

Our main contribution includes proposing the application of model selection mechanisms to select historical AIOps models. We establish the theoretical performance upper bound for model selection mechanisms on historical models through a hypothetical oracle. We are also the first to conduct an empirical study that evaluates historical model selection mechanisms on AIOps solutions. In addition, We share a replication package which includes our code for conducting case study on the studied operation datasets and analyzing the experiment results,[1] so that others in the research community can replicate or extend our work.

The paper is organized as follows. Section 2 presents an overview of prior works in AIOps solution and model selection mechanisms. Section 3 details our case study design and methodology for evaluating model selection mechanisms on historical models in the area of AIOps. Section 4 delivers our experiment results and analysis. Section 5 discusses the limitations of our work and possible threats to validity. Finally, Section 6 concludes our work.

2 Related Work

2.1 Prior research on AIOps solutions

Although huge efforts have been devoted to large-scale software systems like cloud computing to ensure the quality of services, various types of operational incidents (e.g., job termination, hard drive failure, and performance anomalies) are still unavoidable. To ensure the reliability of uninterrupted services, operational incidents must be identified, resolved, and managed in a timely manner, as failing to do so could interrupt service availability and incur

[1] `https://github.com/EmpyreanKnight/suppmaterial-25-yingzhe-AIOpsSelection`

massive financial damage [74]. Prior works have proposed various AIOps solutions for addressing different problems in the operation of large-scale software and systems, including incident prediction [33,30,20,51,8,42,65,14], anomaly detection [22,32], ticket management [67,66], issue diagnosis [38], and self healing [18,19,36,37]. These AIOps solutions can be categorized into two phases that contribute to incident management: 1) *incident perception*, which predicts whether certain types of incidents would occur by learning from the historical operation data; and 2) *incident mitigation*, which remediates the damage from incidents (e.g., automated problem diagnosis) or provides suggestions to domain experts (e.g., incident triage) after the incident occurrences.

2.1.1 Incident perception

Incident perception is a vital step towards early detection of potential incidents and proactive prevention of system failures. Prior works have focused on the identification of various types of incidents by analyzing monitoring data. These approaches can be categorized into two types: failure prediction [20,51,8,42,28,29,65,76,14,33,30,68,1] and anomaly detection [72,71,64,69,2].

Failure prediction. Failure prediction involves forecasting potential system failures before their occurrence by analyzing historical data and identifying patterns that precede failures. For example, El-Sayed et al. [20] and Rosa et al. [51] predict job failures from trace data collected from the Google cloud computing platform. Botezaku et al. [8], Mahdisoltani et al. [42], Li et al. [28,29], and Xu et al. [65] leverage SMART-based monitoring data to build a machine learning pipeline for predicting hard drive failures in large-scale cloud computing platforms. Zhao et al. [76] 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. Similarly, Chen et al. [14] collect and analyze the alert data and its dependencies to predict outages in the whole cloud system. Lin et al. [33] and Li et al. [30] 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 information). Yang et al. [68] propose a novel diffusion model that enhances data quality through data imputation to improve the performance of the downstream failure prediction task in the cloud scenario. Alharthi et al. [1] propose a two-stack transformer-decoder architecture to predict failures as well as the lead times in HPC systems.

Anomaly detection. Anomaly detection aims to detect abnormal system behaviors or patterns that indicate potential issues or failures to help developers and operators uncover system issues and solve anomalies. For instance, Zhang et al. [72] propose a deep-learning-based microservice anomaly detection approach that uses a unified graph representation to describe the complex structure of a trace together with log events embedded in the structure. Zeng et al. [71] propose an actionable performance anomaly alerting approach based on trace data for online service systems. Wu et al. [64] conduct a comprehensive evaluation of seven supervised anomaly detection models with six log representations on four public datasets. Yang et al. [69] propose a semi-supervised approach for detecting log-based anomalies that can stay immune to unstable log data via semantic embedding. Almodovar et al. [2] propose a fine-tuned language model that is robust to log content change, *LogFiT*, for anomaly detection in the HuggingFace ecosystem.

2.1.2 Incident mitigation

Incidents in software systems need to be mitigated in a timely manner. Prior works have focused on triaging [10,11,4,22], diagnosing [75,38,3,25,13, 61,44], and managing issues [36,37,66,67,26,34,32], which benefit the mitigation process.

Triaging. Chen et al. [10] 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. [11] 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. [4] propose *DeCaf*, a Random Forest-based framework to correlate telemetry data with performance regressions. In addition, the detected performance regressions are automatically triaged to the on-site engineering team.

Diagnosing. Zhang et al. [75] propose an ensemble of models to automatically diagnose performance problems. Chen et al. [13] propose *LiDAR*, a deep-learning-based approach to linking similar incidents based on historical information. Luo et al. [38] mine time-series data and event data to discover correlations between them, which could improve the incident diagnosis process. Banerjee et al. [3] discuss challenges in performance diagnosis in a hybrid-cloud enterprise software environment. Jehangiri et al. [25] present techniques to diagnose performance anomalies using time-series datasets.

Managing. Jiang et al. [26] analyze the similarity between incident descriptions and their corresponding troubleshooting guide to facilitate incident management. Lou et al. [36,37] develop a software analytic-based system to resolve the scalability, reliability, and maintainability of data-driven incident management systems. Lim et al. [32] leverage performance metrics to cluster performance issues into recurrent and unknown ones. Xue et al. [66,67] proactively reduce performance tickets by predicting usage series in cloud data centers. Lin et al. [34] propose a data mining-based technique to detect emerging issues (a sudden burst of new issues) by analyzing historical issues.

2.2 Prior work on model selection techniques

Various machine learning models have been proposed for different inference and prediction tasks. However, no existing model can accommodate every type of data and goal. Model selection mechanisms mitigate the challenge by considering a set of candidate models and selecting the most appropriate ones for the prediction task [17]. For example, Liu et al. [35] propose Consistent Relative Confidence (CRC), a label-free model selection method using only unlabeled testing data based on the confidence of candidate models. The authors find that there is a strong positive relationship between consistent relative confidence and correctness among a group of rational volunteers: if there is one candidate who acts more confidently than the others in a consistent way, then its decision tends to be more accurate than the others. Hu et al. [23] propose a labeling-free model selection approach against performance degradation on out-of-distribution data. The main idea is to statistically learn a Bayesian model with the expectation-maximization (EM) algorithm to infer the models' specialty only based on predicted pseudo-labels.

In the context of continual or lifelong learning [47], a great number of candidate models can be available from previous updates, and choosing the right one with the best generalization property for the current task remains a tough challenge. Given a set of candidate models, there is no prior evidence of which model will be capable of solving the targeted task the most effectively [23]. Prior work reports that model selection mechanisms can benefit model performance and robustness when facing concept drift as they leverage the diversity of models to provide robust predictions in dynamic and changing conditions [16,73]. When predicting the test samples, model selection mechanisms can estimate the model performance and rank the models to pick models that most likely to perform best. Research in the AIOps area also suffers from the challenge of concept drift and usually relies on periodical retraining to maintain model performance [30,15,33]. The historical models being discarded may contain valuable information and perform well on specific future periods [48]. However, there is no prior work that systematically examines the performance of model selection mechanisms on historical models in the context of AIOps.

3 Experiment Design

3.1 Case study setup

In order to evaluate different mechanisms of model selection using historical models in the context of AIOps, we perform a case study on three large-scale operation datasets: the Google cluster trace dataset [63], the Backblaze disk stats dataset [24], and the Alibaba GPU cluster trace dataset [21]. We choose to carry out a case study on these three datasets for the following reasons: 1) they are publicly available; 2) they are large-scale datasets and cover relatively long operation periods (i.e., months to years), which enables us to examine model selection mechanisms over the evolution of the data. In addition, prior works have widely studied the first two datasets in particular for predicting job failures on the Google dataset [20,52] and predicting disk failures on the Backblaze dataset [8,65,42]. In this work, we focus on predicting job outcomes (i.e., failure or not) on the Google and Alibaba cluster trace datasets and predicting disk failures on the Backblaze disk stats dataset as done in prior works [40,41,39].

3.1.1 Google cluster trace dataset

The cluster data released by Google in 2011 contains the trace data of a production cluster with about 12K machines in 29 days for 670K jobs and 26M tasks [12]. The data features workload arrives at a cell (i.e., a set of machines that share a common cluster-management system) in the form of jobs. Each job comprises one or more tasks, and each task is scheduled on a single machine. Figure 1a shows the dataset schema and information provided in the Google cluster trace dataset.

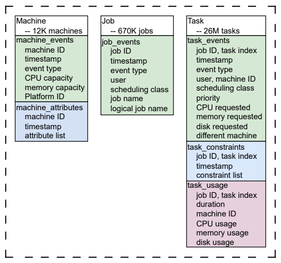
Following prior works [20,52], our goal on the Google cluster data is to predict whether a job will fail or not (i.e., terminated for any reason before successfully completed) using the information at job submission and the monitoring data in the first five minutes of the job execution. In the Google cluster trace dataset, each job has several events, and each is associated with a transit (e.g., submit, schedule, evict, fail, kill, finish, lost, update) among the states (e.g., unsubmitted, pending, running, dead) in the job's lifecycle. We consider a job fails if its final state is "fail", same as in prior works [20,52].

Similarly to El-Sayed et al. [20], we predict job failures using the configuration and temporal features. Configuration features are values determined upon job submission, such as the requested CPU, memory, and disk space. In contrast, temporal features are values that change during a job's execution, such as the mean and standard deviation of CPU, memory, and disk space usage by a job over the first five minutes since job submission.

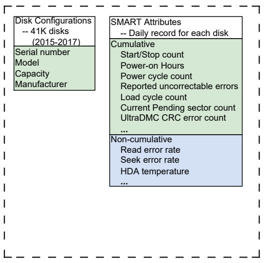
We remove the jobs that are not completed or whose records are lost during execution, as the final states of these jobs are missing. We further remove the jobs that start on the last day (i.e., the 29th day), as these jobs are more likely to remain incomplete before the data cutoff time and cause data collection bias (completed jobs typically last longer than the failed ones). In fact, we observed a much higher job failure rate from the jobs starting on the last day. In addition, we removed jobs that finished in less than five minutes since their submission as they have not generated sufficient metrics for prediction. We also observe that a large proportion of these jobs are failed or terminated right after submission, thus they do not cause significant overhead to the computing resources. In the end, we successfully extracted 627K (out of 670K) job samples from the first 28 days' trace data.

3.1.2 Backblaze disk stats dataset

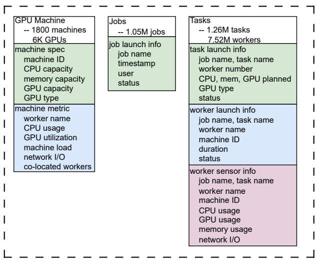
The Backblaze dataset contains the statistics of the hard drives in the Backblaze data center [24]. The dataset contains daily snapshots of operational hard drives in the data center, including drive information (e.g., model, disk capacity) and SMART (Self-Monitoring, Analysis, and Reporting Technology)



(a) Google data schema.



(b) Backblaze data schema.



(c) Alibaba data schema.

Fig. 1: Data schema for our studied datasets. Each colored box represents a data table: a line of the table name followed by lines describing the data fields. For the Google and Alibaba datasets, each table (e.g., machine_events) is one or multiple CSV files containing the fields described in the box. For the Backblaze dataset, the tables represent the logical view, while the physical data is stored as daily snapshots of each disk's attributes.

statistics, where SMART is a manufacturer-implemented system for the monitoring and early detection of errors. Figure 1b shows the dataset schema and information provided in the Backblaze disk stats dataset. The Backblaze disk stats dataset contains hard drive monitoring data collected from 2013 to 2020. Initially, from 2013 to 2014, the trace captured 40 different SMART attributes; then, from 2015 to 2017, the number of SMART attributes increased to 45; starting from the fourth quarter of 2018, 62 different SMART attributes are in the data. Despite the change of monitored attributes, the data type and the monitoring interval (i.e., daily) are kept consistent from 2013 to 2020. We focus on the data collected from 2015 to 2017 because: 1) the subset contains a large number of samples (i.e., over 40M samples), and 2) the subset contains a fixed set of SMART attributes while the data in other periods contains less (before 2015) or more (after 2017) SMART attributes.

Following prior works [42,8], our goal on the Backblaze dataset is to predict hard drive failures (i.e., sector error) within a given future time period (i.e., one week) based on the monitoring data captured during a period of time (i.e., one week) in the past. We consider a disk fails if its "sector error count" SMART attribute increases (i.e., observe sector errors) in the given future time period, same as described in the prior work [42]. The SMART attributes in the Backblaze dataset can be categorized into two types: cumulative attributes whose values are accumulated counts over the disk's lifetime, such as the "reallocated sectors count"; and noncumulative attributes whose values reflect only the current status, such as the "read error rate". Knowing the recent changes in cumulative attributes rather than their raw values might be more insightful. Therefore, we capture both the value change in the past time period and the raw value in the last day of the one-week past window as features for cumulative attributes while only capturing the last day's value for noncumulative attributes. As a result, we collect 11 features from the raw values and 8 features derived from the raw values' differences. The collected features are the same as those used in prior work [42], the most predictive ones selected from all the traced SMART attributes; all 19 features are temporal features. We then collect data samples along a sliding one-week time window and only track the disks that are alive during the whole time window. As a result, we extract 41M samples from the daily snapshots between 2015 and 2017.

3.1.3 Alibaba GPU cluster trace dataset

The GPU cluster trace dataset from Alibaba provides traces of workloads collected from the operation of a large-scale data center [21]. The trace data is collected from runtime information on over 6,500 GPUs across about 1,800 machines in a period of 2 months spanning from July to August of 2020 [62]. The dataset features ML jobs submitted by various users. Once a user submits a job, the job is translated into multiple tasks of different roles. Subsequently, each task is then allocated as instances running on machines. Figure 1c illustrates the trace schema and available information provided in the Alibaba trace dataset. Similar to the Google cluster trace dataset, sensitive fields such as username and job name are desensitized to protect users' privacy.

We define the task as predicting job outcomes using the configuration information and performance metrics available in the first five minutes since job submission [39], similar to our case study on the Google cluster trace dataset. The dataset contains cluster monitoring data for a total of 69 days (around nine weeks). To avoid abnormality (e.g., truncated and untracked jobs) on data samples close to the beginning and end of the trace data, we initiate feature extraction from the fourth day since the trace starts and collect features for a total of 8 weeks. Similar to our handling method on the Google cluster trace dataset, we also removed unfinished jobs and jobs ending in less than five minutes and extracted 701K out of the total of 1.26M jobs.

3.1.4 Dataset segmentation

In this case study, we simulate the real-world scenario where AIOps models are trained on a chunk of initially available data samples and then deployed in the field to predict future samples coming in batches. We follow the approach used in prior studies for updating AIOps models [33,30,65,39] to partition the studied operation datasets. Specifically, we partition each studied dataset into multiple time periods based on the natural time intervals, as described below:

- **Google cluster trace dataset.** We partition the entire 28-day trace data into 28 one-day time periods.
- **Backblaze disk stats dataset.** We partition the entire 3-year monitoring data into 36 one-month time periods.

– **Alibaba GPU cluster trace dataset.** We partition the 2-month GPU trace data into 8 one-week time periods.

For each dataset, we use samples from the first half of the time periods as the initially available training data and the second half of the time periods as the testing data, consecutively coming in by temporal order.

3.1.5 Model training

To ensure that the result of our case study is generalizable, we include a variety of machine learning classifiers that have been used in prior works for predicting disk failures on the Backblaze disk stats dataset and job failures on the Google and Alibaba cluster trace datasets [20,42,8,40,39] in our case study. The list of models we use includes Logistic Regression (LR), Classification and Regression Trees (CART), Random Forest (RF), and Multi-Layer Perceptron Neural Network (NN). We follow the same model configuration as recorded in prior work [51,42,40,39]. We train the models using a sliding window training set with the length fixed to half of the total time periods. Whenever a new testing period concludes and before the testing on the next time period, we train a new sliding window model to maintain model performance against concept drift (i.e., periodical retraining).

The studied operation datasets can be extremely imbalanced, with only 1% job failure in the Google dataset and 0.1% hard drives failures 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. For the Alibaba GPU cluster datasets, we do not applied downsampling to the training samples as the job failure rate is 34.5%. 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. To mitigate the impact of varying feature scales in different time periods on model performance and interpretation, we perform data standardization on each metric in the training dataset by removing the median of the metrics and scaling the metrics according to the quantile range. We apply the same data scaler that was fitted on the training dataset for the respective historical model when predicting testing samples.

Prior works [57,56,54] also suggest that hyperparameter settings can significantly impact the performance of prediction models. Therefore, we tune the hyperparameters of our studied models on the training data using a random search. We choose a random search instead of a grid search for our hyperparameter tuning as using random search is more efficient and can find models that are as good as or better than using a grid search [6].

3.2 Model selection mechanisms

Due to the diverse underlying mechanisms that cause concept drift, the best methods for enhancing models' robustness against it differ across different datasets and shifts [73]. Therefore, we assess the performance of various model selection mechanisms on recycling the previously trained historical models, including two labeling-free model selection mechanisms proposed in prior works [35,23], two model selection mechanisms leveraging temporal adjacency and feature similarity proposed by ourselves, and two variants on the aforementioned mechanisms utilizing the temporal adjacency or similarity. When predicting samples from a target testing period, the model selection mechanisms first estimate the performance of candidate historical models on the testing samples and then rank the models based on the estimation.

We define the problem of model selection on historical models as follows. Given the set of n historical models $S_{\mathcal{M}} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n\}$ and the testing context $\mathcal{C}_t = \{s_1, s_2, \ldots, s_m\}$ on time period t, where samples in $\mathcal{C}_t$ are unlabeled[2], the task is to estimate the rank r of the historical models in $S_{\mathcal{M}}$ according to their expected performance on the whole testing context $\mathcal{C}_t$ with as small rank error as possible. Historical context $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{t-1}$ and their respective labels $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{t-1}$ are also available for the model selection mechanisms.

In our case study, we empirically evaluate the following model selection mechanisms on historical models.

3.2.1 Temporal adjacency based selection mechanism (TBM)

Prior works utilize the temporal adjacency in training and estimation of model performance in the environment of rapidly changing data distribution [70]. We consider the samples from the last period of time to be the most similar to the testing samples in terms of data distribution and estimate the performance of historical models with their performance on the last available period. We also add a revised version (rTBM) that further utilize the temporal adjacency: when the highest ranked model is from the second latest historical time periods, we assume the model from the latest historical time period (which was not tested due to the concern of data leakage) would be better and bring it to the top of the ranking.

3.2.2 Similarity based selection mechanism (SBM)

Aside from utilizing temporal adjacency to approximate the most similar historical samples to the testing samples, we also devise a model selection mechanism to find the most similar samples from historical time periods to the testing samples directly using distance measurements. Hausdorff distance [7]

[2] For example, for the job failure prediction task, the samples in $\mathcal{C}_t$ contain information about the characteristics of the jobs (e.g., configurations, early running status), and we need to predict the job outcomes which are unknown (i.e., unlabeled).

measures how far two subsets of a metric space are from each other and represents the greatest of all the distances from a point in one set to the closest point in the other set. We calculate the Hausdorff distance between the samples in the testing period and the samples from each of the historical periods starting from the ending period of the first training window and find the most similar historical period to validate the historical models. We then rank the historical models based on the performance of the chosen time period (i.e., the most similar historical period to the testing period). We further place the ranking of the historical models without data leakage (i.e., no intersection between the training window and the chosen time period) above the models with data leakage to mitigate the issue. We choose to use the Hausdorff distance as other distance measurements cannot be applied to multivariate distributions with different amounts of observations (e.g., Hellinger distance) or are computationally intensive (e.g., Wasserstein metric). We also provide a revised version (rSBM) that ignores the data leakage issue and ranks the historical models altogether.

3.2.3 CRC model selection mechanism (CRC)

CRC is a labeling-free selection mechanism that uses the predicted probabilities on the testing samples to estimate model performance [35]. It assumes that a candidate model with higher confidence should yield more accurate prediction results and better performance. The CRC mechanism first obtains the predicted probabilities of testing samples from each historical model. It then ranks the models based on the average confidence calculated from the highest predicted probabilities among all the classes on each sample from the testing period.

3.2.4 LaF model selection mechanism (LaF)

LaF is also a labeling-free model selection mechanism that ranks the candidate models by maximizing the expectations of each model on consensus-decided pseudo-labels of the testing samples [23]. The LaF mechanism first uses majority voting on the prediction results from candidate models to estimate the pseudo-label of the testing samples. It then removes the testing samples in which all candidate models agree on the prediction results (i.e., where a unanimous agreement occurs) as these samples possess no discriminability against the candidates. LaF then employs the expectation-maximization algorithm on the trimmed-down testing set to optimize the estimation of candidate model capabilities. Finally, it uses the final estimation to rank the candidate models.

3.2.5 Conventional baselines

We consider two conventional baselines in our case study: stationary model and periodical retraining. The two baselines are conventional model updating strategies that are widely used in prior AIOps works [20, 51, 8, 42, 14, 67, 30, 15, 33]. A stationary model is trained on the initial training data and never updates while a periodical retraining mechanism updates the model with sliding window training samples each time data from a new period becomes available. The stationary model can be seen as a model selection mechanism that always selects the initially trained model while the periodical retraining can be seen as a model selection mechanism that always selects the newest model.

3.2.6 Hypothetical oracle

We also design a hypothetical oracle that exploits the true labels of samples from the target time period to rank and select the historical models. During the inference of target samples, their label are not available yet hence the oracle shows a theoretical performance only. The hypothetical oracle first estimates the performance of candidate historical models with testing samples and the yet-to-come labels then ranks the candidates with the prediction performance estimation. The oracle serves as a *theoretical optimal* performance indicator for historical model selection mechanisms.

3.3 Evaluation of model selection mechanisms

We measure the performance of model selection mechanisms from three aspects: prediction performance, ranking performance, and ranking consistency. For the prediction performance, we measure how well the top model chosen by each model selection mechanism can predict the incidents on testing samples. For the ranking performance, we measure how similar the rankings from model selection mechanisms are to the actual model ranking by their performance on the testing samples. We also measure the consistency of model rankings from the same model selection mechanism in different runs (i.e., with different random seeds).

3.3.1 Prediction performance

We evaluate the performance of our models using the Area Under the Receiver Operating Characteristic Curve (AUC) metric, which is a standard and widely used metric for evaluating machine learning models. AUC measures model performance by calculating the area under the curve of true positive rate (TPR) against false positive rate (FPR) at different classification thresholds. Prior work [55] shows that one should use threshold-independent metrics such as 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 as the performance indicator of prediction results on the testing samples. We measure the AUC performance of each model selection mechanism on each testing period (i.e., from period $l/2+1$ to period l for the l time periods). We repeat the experiment 100 times with different random seeds to mitigate the performance fluctuation.

To statistically compare the performance of different combinations of model update strategies and model choices, we apply the Scott-Knott test. The Scott-Knott test is a hierarchical clustering method [53] that groups observations into statistically distinct clusters. The observations within a group have no statistically significant difference (i.e., p-value ≥ 0.05), while the observations in different groups have a statistically significant difference (i.e., p-value < 0.05). In our case, the observations are the values of AUC from 100 repeated experiments with different random seeds.

3.3.2 Ranking Performance

Model selection mechanisms rank the candidate models by their estimated performance on the testing samples. To compare the similarity between the ranking of each model selection mechanism and the ground truth (i.e., ranking of models based on the true AUC performance on the testing samples), we use the following metrics:

Kendall's τ. Kendall's τ measures the non-parametric rank correlation between two rankings for evaluating the overall agreement. Kendall's τ ranges from 0 (no agreement) to 1/-1 (perfect agreement/disagreement). We focus on only the agreement of model rankings and use the same interpretation schema from prior work [50] to interpret Kendall's *tau* in our study:

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

In our case study, Kendall's τ measures the *overall agreement* between rankings from model selection mechanisms and the ranking from the hypothetical oracle.

Jaccard similarity coefficient J_k. The Jaccard similarity coefficient J_k evaluates the similarity between the top-k model sets generated by the two rankings. Prior work usually consider values of $k = 3, 5, 10$ [23,35,43] to evaluate the ranking performance of model selection mechanisms on different cutoff points. Given n historical models $S_{\mathcal{M}} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ and two ranking functions r_1 and r_2 on models $S_{\mathcal{M}}$. The Jaccard similarity coefficient on top-k models J_k is defined as:

$$J_k = \frac{|\cap_i \{\mathcal{M}_j | r_i(\mathcal{M}_j) \leq k\}|}{|\cup_i \{\mathcal{M}_j | r_i(\mathcal{M}_j) \leq k\}|}$$

where $r_i(\mathcal{M}_j) \in \{1, ..., m\}$, $i \in \{1, 2\}$, and $j \in \{1, ..., m\}$. A large J_k indicates a higher degree of similarity between two rankings. In our case study, the Jaccard similarity coefficient measures the *agreement on top-k model candidates* between the rankings from model selection mechanisms and the ranking from the hypothetical oracle. Since the Alibaba dataset only contains four testing periods (i.e., 4 historical models at maximum) while the prediction performance effectively measured the performance when $k = 1$. Therefore, in our case study, we only report the Jaccard similarity with $k = 3$ (i.e., J_3). We include the results for other cutoff points in our replication package.

3.3.3 Ranking consistency

To measure the consistency of ranking results provided by each model selection mechanism on each testing period, we calculate Kendall's W. Kendall's W is a non-parametric measure of the agreement among multiple rankings, ranging from 0 (no agreement) to 1 (complete agreement). In our case study, we calculate Kendall's W among the rankings from 100 runs in each of the combinations of testing period, model, and dataset. We use the same interpretation schema that is applied to Kendall's τ to evaluate the degree of agreement.

We calculate these ranking performance metrics on each testing period starting from the second, as the first testing period contains only one historical model, which makes the rankings trivial. We also apply the Scott-Knott test to group the results from the 100 repeated experiments, providing a statistical basis for comparison.

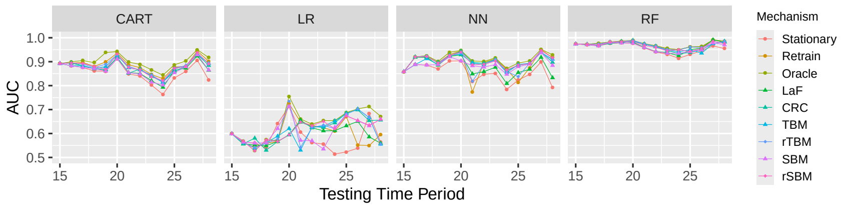
4 Experiment Results

We organize the experimental results along the evaluation of model selection mechanisms in the prediction performance of the top-ranked models, the ranking performance of model selection mechanisms when compared with the hypothetical oracle, and the ranking consistency inside each of the model selection mechanisms. The three evaluation aspects correspond to the three RQs we aim to address.

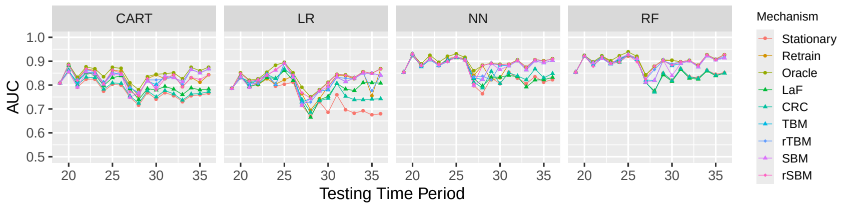
4.1 RQ1: How well can the model selection mechanisms achieve optimal performance for AIOps solutions?

Figure 2 shows the AUC performance of each model selection mechanism using the top-ranked models in each testing period. Figure 3 further groups the AUC performance of each model selection mechanism using the Scott-Knott ranking test.

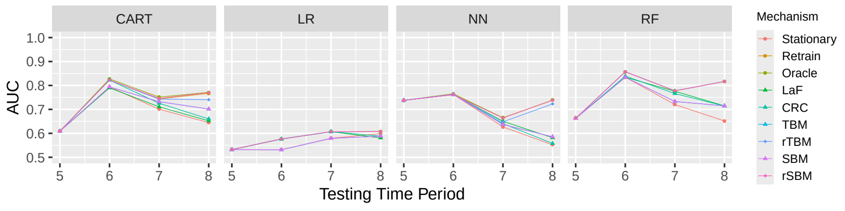
The oracle shows higher performance than the periodical retraining baseline in many cases, indicating possible improvements over periodical retraining for model selection mechanisms. We observe higher AUC performance on the oracle baseline than the periodical retraining baseline in three models (i.e., NN, CART, and LR) from the Google dataset and all four models (i.e., RF, NN, CART, and LR) from the Backblaze datasets. The observed performance differences are also statistically significant. As the model selection mechanisms can choose from all historical models rather than being limited to the most recent one, these findings confirm the theoretical potential of model selection mechanisms in outperforming periodical retraining for maintaining AIOps solutions in the field. However, we do not observe a significant performance difference between the oracle and periodical retraining on the Alibaba dataset. This situation may result from the limited



(a) Google



(b) Backblaze

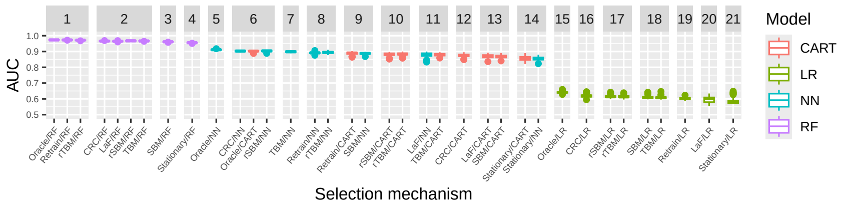


(c) Alibaba

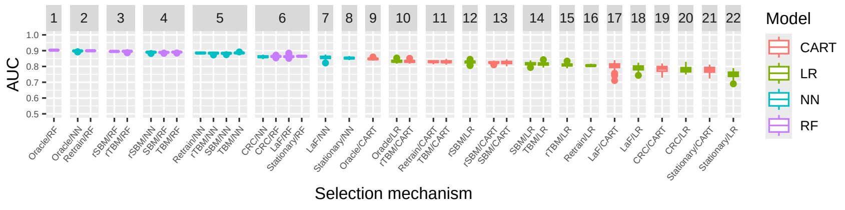
Fig. 2: The average AUC performance of model selection mechanisms in each testing period.

number of available historical models due to the relatively smaller volume of time periods and shorter trace data duration, which restricts concept drift from emerging.

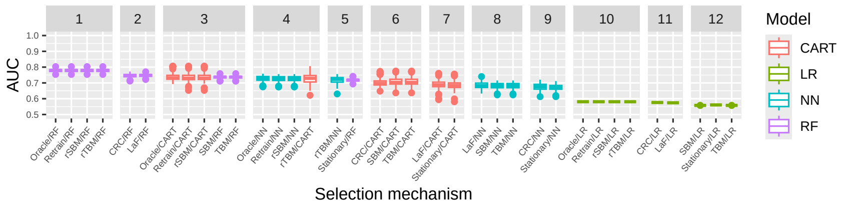
Several model selection mechanisms achieve higher performance than the periodical retraining baseline. We observe that several model selection mechanisms achieve statistically better performance than the periodical retraining baseline on the Google and Backblaze datasets. On the Google dataset, the CRC, rSBM, and TBM model selection mechanisms with the NN model, as well as the CRC, rSBM, rTBM, SBM, and TBM mechanisms with the LR model, achieve statistically better performance than the periodical retraining baseline. On the Backblaze dataset, the rSBM mechanism with the NN model, the rTBM mechanism with the CART model, and the rSBM, SBM, rTBM, and TBM mechanisms with the LR model all achieve statistically better performance than the periodical retraining baseline. However, we have not observed statistical performance differences on the Alibaba dataset due to the reasons mentioned above.



(a) Google



(b) Backblaze



(c) Alibaba

Fig. 3: Scott-Knott test results of the AUC performance from different multi-model selection mechanisms.

The CRC, rSBM, and TBM mechanisms on the Google dataset, and the rSBM and rTBM mechanisms on the Backblaze dataset, show a generally better performance than other model selection mechanisms. On the Google dataset, the CRC, rSBM, and TBM mechanisms achieve better performance than the periodical retraining baseline with NN and LR models. In addition, rTBM shows performance comparable to the periodical retraining baseline in RF, NN, and LR models. On the Backblaze dataset, the rSBM and rTBM mechanisms achieve better performance than the periodical retraining baseline on two models each (NN and LR, CART and LR, respectively). The SBM and TBM mechanisms also achieve comparable or higher performance to the periodical retraining on most models. Although no model selection mechanism explicitly achieves superior performance compared to the periodical retraining baseline on the Alibaba dataset, rSBM and rTBM mechanisms present comparable results in most scenarios.

The revised versions of the TBM and SBM mechanisms tend to have a better performance than the original ones. For the TBM mechanism, we observe that the revised version (i.e., rTBM) achieve better performance than the original one with three models (i.e., RF, CART, and LR) on the Google dataset, two models (i.e., RF and CART) on the Backblaze dataset, and all four models on the Alibaba dataset. The revised version of TBM leverages the temporal adjacency to extend estimations to the most recent historical model, hence increasing the chance of selecting the best-performing model as the top-ranked choice. For the SBM mechanism, the revised version (i.e., rSBM) achieves better performance than the original one with all four models on the Google dataset, three models (i.e., RF, NN, and LR) on the Backblaze dataset, and all four models on the Alibaba dataset. We surmise that the rSBM mechanism has a greater chance of selecting more recent historical models when ignoring the intersection with training samples. Such behavior further shows the weight of temporal adjacency when selecting historical models.

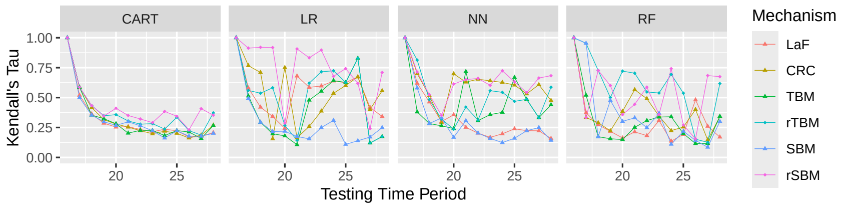
The rTBM mechanism can achieve statistically comparable performance to the oracle baseline in some scenarios. We observe that the rTBM mechanism achieves comparable performance with the RF model on the Google dataset. Although several model selection mechanisms also achieve similar performance to the oracle baseline on the Alibaba dataset, we do not consider these cases valid as the performance groupings are insufficiently discriminative due to the short duration.

Summary of RQ1

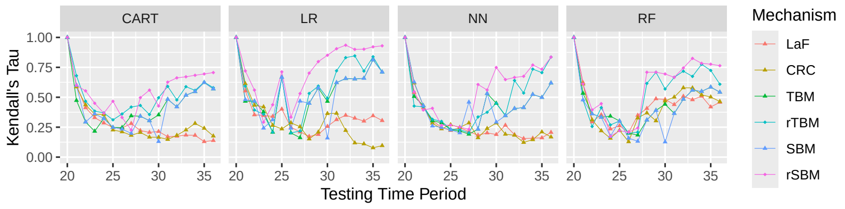
In terms of prediction performance using the top-ranked model, we observe several model selection mechanisms achieve higher performance than the periodical retraining baseline, with the rTBM mechanism achieving statistically comparable performance to the oracle baseline in some scenarios. However, there is still a gap between the theoretical upper bound performance indicated by the hypothetical oracle and the performance of the existing model selection mechanisms.

4.2 RQ2: How well can the model selection mechanisms achieve optimal model ranking for AIOps solutions?

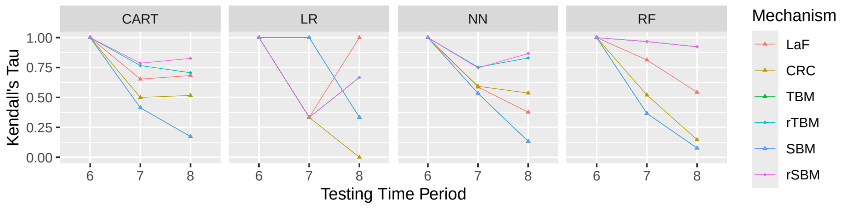
We measure the similarity between the rankings from model selection mechanisms and the oracle baseline using Kendall's τ and Jaccard similarity coefficient as indicators of ranking performance. We also calculate Kendall's W among the rankings from the same model selection mechanism in each testing period to measure the consistency of rankings. For Kendall's τ, Figure 4 shows the Kendall's τ between the rankings on each model selection mechanism and the oracle baseline in each testing period and Figure 5 further groups the performance of each model selection mechanism using the Scott-Knott ranking



(a) Google



(b) Backblaze

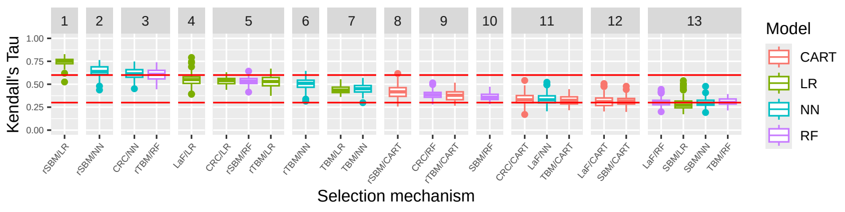


(c) Alibaba

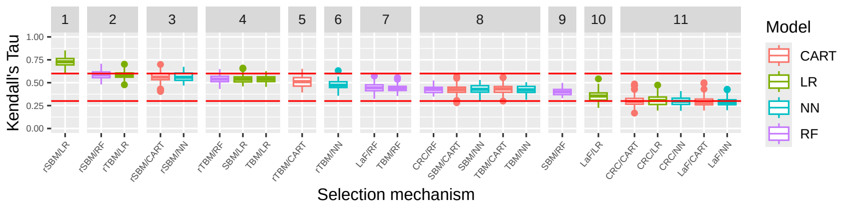
Fig. 4: The average Kendall's τ correlation between model selection mechanisms and the oracle ranking in each testing period.

test. Similarly, for the Jaccard similarity coefficient, Figure 6 shows the Jaccard similarity coefficient between the rankings on each model selection mechanism and the oracle baseline in each testing period and Figure 7 further groups the performance of each model selection mechanism using the Scott-Knott ranking test.

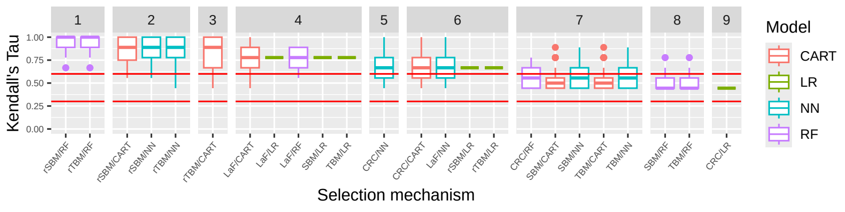
The rSBM model selection mechanism provides rankings most aligned with the oracle baseline across datasets and models. Figure 4 and Figure 5 show that the rSBM model selection mechanism consistently achieves the highest Kendall's τ, which indicates strong ranking similarity to the oracle. Specifically, rSBM is ranked within the top statistical group for three (i.e., LR, NN, and CART) of the four models on the Google dataset. Also, rSBM achieves the highest statistical group ranking for all four models on the Backblaze dataset, and for three (i.e., RF, CART, and NN) models on the Alibaba dataset. The rTBM model selection mechanism similarly ranks among the top group in terms of Kendall's τ. For example, rTBM achieves the



(a) Google



(b) Backblaze

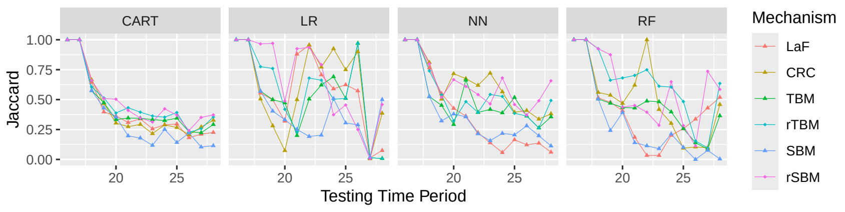


(c) Alibaba

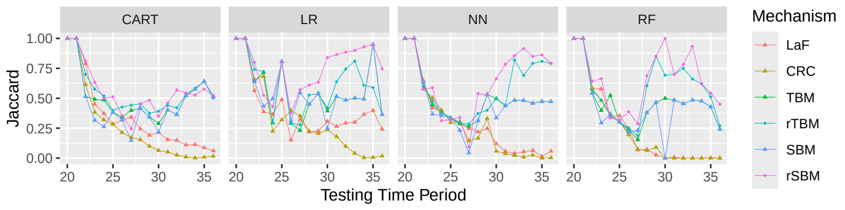
Fig. 5: Scott-Knott test results of the Kendall's τ correlation from different multi-model selection mechanisms. The horizontal lines indicate the threshold for interpreting ranking agreement.

top group for the RF model on the Google dataset, for the LR model on the Backblaze dataset, and for the RF and NN models on the Alibaba dataset.

The rSBM and rTBM model selection mechanisms perform well in terms of top-k agreement. As shown in Figure 6 and Figure 7, we observe similar performance for the rSBM and rTBM model selection mechanisms based on Jaccard similarity coefficient with $k = 3$. On the Google dataset, the rSBM model selection mechanism is within the top statistical performance group for LR and CART, while rTBM achieves the highest statistical group ranking for the RF model. For the Backblaze dataset, rSBM maintains top-level performance across all four models, while rTBM achieves the top statistical group ranking for the LR and CART models. We omit the Alibaba dataset from this analysis due to the limited number (i.e., four) of testing periods, which makes Jaccard similarity analysis impractical (i.e., fewer historical models than required by the selected top-k threshold, resulting in trivial or total agreement for early periods).

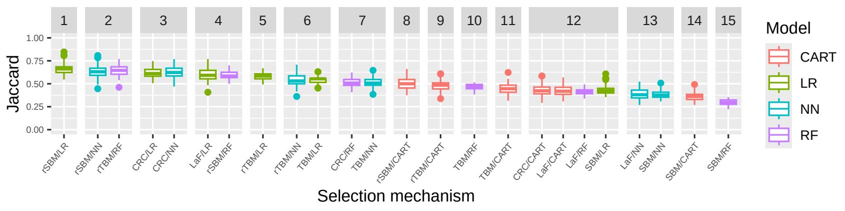


(a) Google

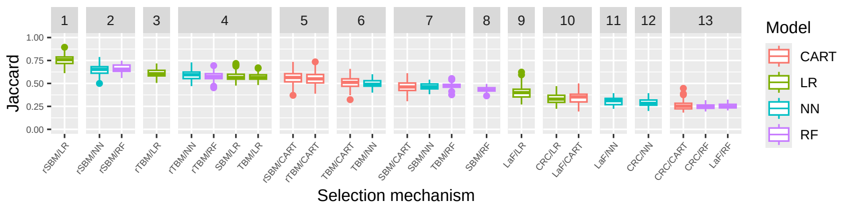


(b) Backblaze

Fig. 6: The Jaccard similarity coefficient ($k = 3$) between model selection mechanisms and the oracle ranking in each testing period.



(a) Google

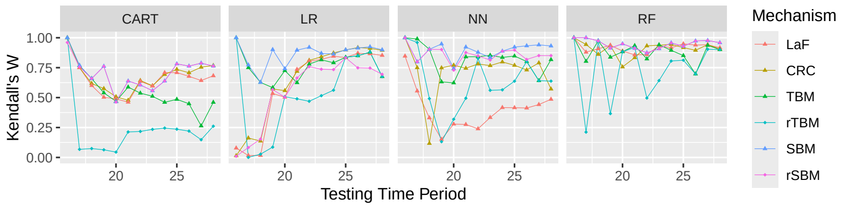


(b) Backblaze

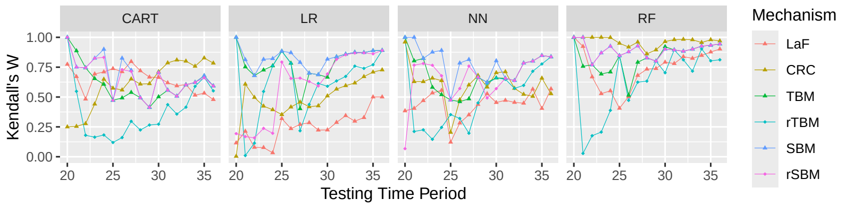
Fig. 7: Scott-Knott test results of the Jaccard similarity coefficient ($k = 3$) from different multi-model selection mechanisms.

Summary of RQ2

We measure the similarity between the rankings from model selection mechanisms and the oracle baseline using Kendall's τ and Jaccard similarity coefficient J_3 as indicators of ranking performance. Our results find that the rSBM model selection mechanism provides rankings most aligned with the oracle baseline across datasets and models, while the rSBM and rTBM model selection mechanisms perform well in terms of the agreement on the top-3 performing models.

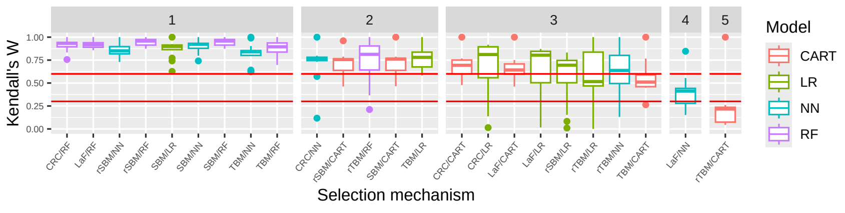


(a) Google

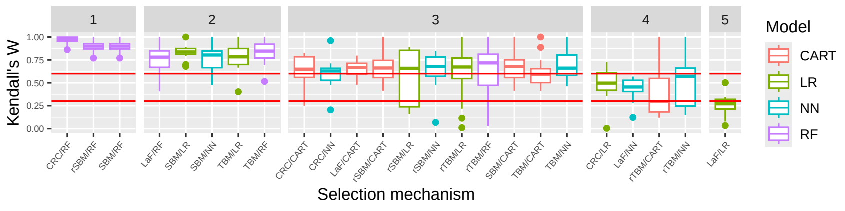


(b) Backblaze

Fig. 8: The Kendall's W correlation among rankings from the same model selection mechanism in each testing period.



(a) Google



(b) Backblaze

Fig. 9: Scott-Knott test results of the Kendall's W correlation from different multi-model selection mechanisms. The horizontal lines indicate the threshold for interpreting ranking agreement.

4.3 RQ3: How stable are the model ranking results achieved by the model selection mechanisms?

Figure 8 shows the Kendall's W among the rankings from each model selection mechanism in each testing period, and Figure 9 further groups the Kendall's W of each model selection mechanism using the Scott-Knott ranking test.

Overall, most model selection mechanisms achieve robust consistency with strong agreement among rankings across repeated runs. As shown in Figure 8 and Figure 9, we observe strong agreement (i.e., Kendall's W > 0.6) for nearly all combinations of model selection mechanisms and models on the Google dataset, with a few combinations demonstrating moderate or weak agreement. The rTBM model selection mechanism for the LR model, TBM for the CART model, and LaF for the NN model achieve moderate agreement, while rTBM for the CART model demonstrates weak agreement. On the Backblaze dataset, the majority of combinations also show strong consistency, with four model and selection mechanism combinations showing moderate agreement, and one combination showing weak agreement (i.e., the LaF mechanism on the LR model). For the Alibaba dataset, we omit Kendall's W analysis due to the limited number of historical models and testing periods available.

Summary of RQ3

We evaluate the ranking consistency inside the model selection mechanism in each testing period. In other words, we measure how the ranking provided by the same model selection mechanism with the same configuration varies from time to time to assess the trustworthiness of the rankings. We observe that most model selection mechanisms achieve robust consistency with strong agreement among rankings across repeated runs, with the LaF and rTBM model selection mechanisms showing weak agreement among multiple runs in some cases.

5 Threats to Validity

5.1 External Validity

One possible threat to external validity is that we only examine the multi-model selection mechanisms on a limited number of datasets. To address this validity, we choose three large-scale and publicly available datasets that represent diverse real-world scenarios (Google cluster trace, Backblaze disk stats, and Alibaba GPU cluster trace). These three datasets have been widely used in prior AIOps research [20,52,8,65,42].

Another potential threat arises from the choice of machine learning models. We employ four representative machine learning models (i.e., Logistic Regression, CART, Random Forest, and Neural Network) that have been used on the studied datasets from prior works [20,42,8,40,39] to ensure fair comparison and generalizability of our results. However, it remains unclear whether our findings directly extend to other types of models or architectures that are not studied in this work.

In addition, the choice of model selection mechanisms could be a potential threat. To address this, we include six model selection mechanisms: two labeling-free mechanisms proposed in prior works [35,23], as well as our proposed TBM and SBM variants (and their revised versions) leveraging temporal adjacency and feature similarity. These selection mechanisms ensure our study covers diverse approaches.

5.2 Internal Validity

One possible threat to internal validity concerns the measurement of performance indicators (i.e., AUC for prediction performance, Kendall's τ and Jaccard correlation for ranking performance, and Kendall's W for ranking consistency). Considering various factors may impact the model performance, we repeat the experiment for 100 runs and control the randomness during each run. The choice of time period partition may also pose a threat to internal validity. We mitigate this threat by using the widely accepted natural time period partition as done in prior works [65,30].

Furthermore, dataset preprocessing steps such as the downsampling strategy applied for handling class imbalance and data standardization may influence results. We adopt best practices widely used in prior empirical research [55,20,40]. Specifically, we follow natural time intervals as recommended in prior AIOps research [65,30], use standard approaches for handling data imbalance (downsampling the majority classes), and apply preprocessing steps such as data scaling according to median-quantile ranges.

5.3 Construct Validity

A threat to construct validity concerns our model training process, as the configuration and parameter settings may impact the experiment results. In order to mitigate this threat to construct validity, we use automated hyperparameter tuning to optimize the configuration of the hyperparameters, which is a widely used practice in the development of machine learning models.

There may be other potential threats concerning our model training process. We use the same features as in prior works for the Google and Backblaze operation datasets and the same types of base models (except for the online learning approaches) have been applied in the prior works [52,20,8,42] to reflect the training process in AIOps solutions. Future work that evaluates our study by considering other modeling processes, e.g., using different features, different ML libraries, or a different re-sampling approach, could benefit our study.

6 Conclusion

In this work, we study the model selection mechanisms on historical models in the context of AIOps through a case study on three large-scale operation datasets and six types of model selection mechanisms. We empirically evaluate the prediction performance when choosing the top-ranked model using model selection mechanisms, the ranking agreement between rankings from the model selection mechanisms and the theoretical oracle, and the ranking consistency inside each model selection mechanism in multiple runs. Our findings suggest the temporal adjacency based selection mechanism tends to have a better performance than other model selection mechanisms and prevails in AUC performance than the periodical retraining. We also find that the rSBM mechanism tend to have the most similar ranking when compared with the oracle ranking. Future work may consider utilizing the ranking from rSBM mechanism further in scenarios like selecting models for time-based ensemble models considering its high accuracy. We also suggest future research to devise model selection mechanisms that can achieve closer performance to the theoretical optimal as our results indicate there is still a gap in the performance between the current model selection mechanisms and the theoretical upper bound.

Declarations

Funding

Not applicable.

Ethical Approval

This study does not involve human participants or animals.

Informed Consent

Not applicable. No human subjects were involved in this study.

Author Contributions

– Yingzhe Lyu: Conceptualization, Data Collection, Methodology, Data Analysis, Writing – Original Draft.

– Hao Li: Methodology, Data Validation, Writing – Review & Editing.
– Heng Li: Supervision, Writing – Review & Editing, Conceptual Guidance, Research Direction.
– Ahmed E. Hassan: Supervision, Research Direction.

Data Availability Statement

The dataset, experiment code, and experiment results of this study are available in our replication package.[3]

Conflict of Interest

The authors declared that they have no known competing interests or personal relationships that could have (appeared to) influenced the work reported in this article.

Clinical Trial Number in the Manuscript

Not applicable.

References

1. Alharthi, K.A., Jhumka, A., Di, S., Gui, L., Cappello, F., McIntosh-Smith, S.: Time machine: Generative real-time model for failure (and lead time) prediction in HPC systems. In: 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, DSN 2023, Porto, Portugal, June 27-30, 2023, pp. 508–521. IEEE (2023)
2. Almodovar, C., Sabrina, F., Karimi, S., Azad, S.A.: Logfit: Log anomaly detection using fine-tuned language models. IEEE Trans. Netw. Serv. Manag. **21**(2), 1715–1723 (2024)
3. Banerjee, A., Chen, C., Hung, C., Huang, X., Wang, Y., Chevesaran, R.: Challenges and experiences with mlops for performance diagnostics in hybrid-cloud enterprise software deployments. In: 2020 USENIX Conference on Operational Machine Learning, OpML 2020, July 28 - August 7, 2020 (2020)
4. Bansal, C., Renganathan, S., Asudani, A., Midy, O., Janakiraman, M.: Decaf: diagnosing and triaging performance issues in large-scale cloud services. In: ICSE-SEIP 2020: 42nd International Conference on Software Engineering, Software Engineering in Practice, Seoul, South Korea, 27 June - 19 July, 2020 (2020)
5. Bendimerad, A., Remil, Y., Mathonat, R., Kaytoue, M.: On-premise AIOps infrastructure for a software editor SME: an experience report. In: S. Chandra, K. Blincoe, P. Tonella (eds.) Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023, pp. 1820–1831. ACM (2023)
6. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(10), 281–305 (2012)
7. Birsan, T., Tiba, D.: One hundred years since the introduction of the set distance by dimitrie pompeiu. In: System Modeling and Optimization: Proceedings of the 22nd IFIP TC7 Conference held from July 18–22, 2005, in Turin, Italy 22, pp. 35–39. Springer (2006)

[3] `https://github.com/EmpyreanKnight/suppmaterial-25-yingzhe-AIOpsSelection`

8. Botezatu, M.M., Giurgiu, I., Bogojeska, J., Wiesmann, D.: Predicting disk replacement towards reliable data centers. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pp. 39–48 (2016)
9. Chaudhary, G., Mebratu, D., Lewis, B., Khanna, R., Jin, J., Hossain, M.: Monitoring workload performance in noisy neighborhoods using performance monitoring units. In: IEEE/ACM International Workshop on Cloud Intelligence & AIOps, AIOps@ICSE 2023, Melbourne, Australia, May 15, 2023, pp. 14–22. IEEE (2023)
10. Chen, J., He, X., Lin, Q., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y., Zhang, D.: Continuous incident triage for large-scale online service systems. In: 34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019 (2019)
11. Chen, J., Zhang, S., He, X., Lin, Q., Zhang, H., Hao, D., Kang, Y., Gao, F., Xu, Z., Dang, Y., Zhang, D.: How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In: 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020 (2020)
12. Chen, X., Lu, C.D., Pattabiraman, K.: Failure prediction of jobs in compute clouds: A google cluster case study. In: Proceedings of the 2014 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW '14, pp. 341–346 (2014)
13. Chen, Y., Yang, X., Dong, H., He, X., Zhang, H., Lin, Q., Chen, J., Zhao, P., Kang, Y., Gao, F., Xu, Z., Zhang, D.: Identifying linked incidents in large-scale online service systems. In: ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020 (2020)
14. Chen, Y., Yang, X., Lin, Q., Zhang, H., Gao, F., Xu, Z., Dang, Y., Zhang, D., Dong, H., Xu, Y., Li, H., Kang, Y.: Outage prediction and diagnosis for cloud service systems. In: Proceedings of the 2019 World Wide Web Conference, WWW '19, pp. 2659–2665 (2019)
15. Dang, Y., Lin, Q., Huang, P.: AIOps: Real-world challenges and research innovations. In: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, ICSE-Companion '19, pp. 4–5 (2019)
16. Diffenderfer, J., Bartoldson, B.R., Chaganti, S., Zhang, J., Kailkhura, B.: A winning hand: Compressing deep networks can improve out-of-distribution robustness. In: M. Ranzato, A. Beygelzimer, Y.N. Dauphin, P. Liang, J.W. Vaughan (eds.) Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pp. 664–676 (2021)
17. Ding, J., Tarokh, V., Yang, Y.: Model selection techniques: An overview. IEEE Signal Process. Mag. **35**(6), 16–34 (2018). DOI 10.1109/MSP.2018.2867638. URL `https://doi.org/10.1109/MSP.2018.2867638`
18. Ding, R., Fu, Q., Lou, J.G., Lin, Q., Zhang, D., Shen, J., Xie, T.: Healing online service systems via mining historical issue repositories. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ASE '12, pp. 318–321 (2012)
19. Ding, R., Fu, Q., Lou, J.G., Lin, Q., Zhang, D., Xie, T.: Mining historical issue repositories to heal large-scale online service systems. In: Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN '14, pp. 311–322 (2014)
20. El-Sayed, N., Zhu, H., Schroeder, B.: Learning from failure across multiple clusters: A trace-driven approach to understanding, predicting, and mitigating job terminations. In: Proceedings of the 37th IEEE International Conference on Distributed Computing Systems, ICDCS '17, pp. 1333–1344 (2017)
21. Group, A.: Alibaba cluster trace program. `https://github.com/alibaba/clusterdata` (2021)
22. He, S., Lin, Q., Lou, J.G., Zhang, H., Lyu, M.R., Zhang, D.: Identifying impactful service system problems via log analysis. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '18, pp. 60–70 (2018)

23. Hu, Q., Guo, Y., Xie, X., Cordy, M., Papadakis, M., Traon, Y.L.: Laf: Labeling-free model selection for automated deep neural network reusing. ACM Trans. Softw. Eng. Methodol. **33**(1), 25:1–25:28 (2024)
24. Inc., B.: Backblaze hard drive stats. Backblaze B2 Cloud Storage (2020). Posted at `https://www.backblaze.com/b2/hard-drive-test-data.html`
25. Jehangiri, A.I., Yahyapour, R., Wieder, P., Yaqub, E., Lu, K.: Diagnosing cloud performance anomalies using large time series dataset analysis. In: 2014 IEEE 7th International Conference on Cloud Computing, Anchorage, AK, USA, June 27 - July 2, 2014 (2014)
26. Jiang, J., Lu, W., Chen, J., Lin, Q., Zhao, P., Kang, Y., Zhang, H., Xiong, Y., Gao, F., Xu, Z., Dang, Y., Zhang, D.: 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, Virtual Event, USA, November 8-13, 2020, ESEC/FSE '20 (2020)
27. Li, H., Chen, T.H.P., Hassan, A.E., Nasser, M., Flora, P.: Adopting autonomic computing capabilities in existing large-scale systems: An industrial experience report. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '18, pp. 1–10 (2018)
28. Li, J., Ji, X., Jia, Y., Zhu, B., Wang, G., Li, Z., Liu, X.: Hard drive failure prediction using classification and regression trees. In: 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014 (2014)
29. Li, J., Stones, R.J., Wang, G., Liu, X., Li, Z., Xu, M.: Hard drive failure prediction using decision trees. Reliab. Eng. Syst. Saf. (2017)
30. Li, Y., Jiang, Z.M., Li, H., Hassan, A.E., He, C., Huang, R., Zeng, Z., Wang, M., Chen, P.: Predicting node failures in an ultra-large-scale cloud computing platform: An AIOps solution. ACM Transactions on Software Engineering and Methodology **29**(2), 1–24 (2020)
31. Liao, L., Eismann, S., Li, H., Bezemer, C.P., Costa, D.E., van Hoorn, A., Shang, W.: Early Detection of Performance Regressions by Bridging Local Performance Data and Architectural Models . In: 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE), pp. 317–329. IEEE Computer Society, Los Alamitos, CA, USA (2025)
32. Lim, M.H., Lou, J.G., Zhang, H., Fu, Q., Teoh, A.B.J., Lin, Q., Ding, R., Zhang, D.: Identifying recurrent and unknown performance issues. In: Proceedings of the 2014 IEEE International Conference on Data Mining, ICDM '14, pp. 320–329 (2014)
33. Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K., Xu, Y., Lou, J.G., Li, C., Wu, Y., Yao, R., et al.: Predicting node failure in cloud service systems. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '18, pp. 480–490 (2018)
34. Lin, Q., Lou, J., Zhang, H., Zhang, D.: idice: problem identification for emerging issues. In: Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 (2016)
35. Liu, B.: Consistent relative confidence and label-free model selection for convolutional neural networks. In: 2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML), pp. 375–379 (2022)
36. Lou, J., Lin, Q., Ding, R., Fu, Q., Zhang, D., Xie, T.: Software analytics for incident management of online services: An experience report. In: Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering, ASE '13, pp. 475–485 (2013)
37. Lou, J.G., Lin, Q., Ding, R., Fu, Q., Zhang, D., Xie, T.: Experience report on applying software analytics in incident management of online service. Automated Software Engineering **24**(4), 905–941 (2017)
38. Luo, C., Lou, J.G., Lin, Q., Fu, Q., Ding, R., Zhang, D., Wang, Z.: Correlating events with time series for incident diagnosis. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 1583–1592 (2014)

39. Lyu, Y., Li, H., Jiang, Z.M.J., Hassan, A.E.: On the model update strategies for supervised learning in aiops solutions. ACM Trans. Softw. Eng. Methodol. **33**(7), 184:1–184:38 (2024)
40. Lyu, Y., Li, H., Sayagh, M., Jiang, Z.M.J., Hassan, A.E.: An empirical study of the impact of data splitting decisions on the performance of aiops solutions. ACM Transactions on Software Engineering and Methodology (2021)
41. Lyu, Y., Rajbahadur, G.K., Lin, D., Chen, B., Jiang, Z.M.J.: Towards a consistent interpretation of aiops models. ACM Trans. Softw. Eng. Methodol. **31**(1), 16:1–16:38 (2022)
42. Mahdisoltani, F., Stefanovici, I., Schroeder, B.: Proactive error prediction to improve storage system reliability. In: Proceedings of the 2017 USENIX Annual Technical Conference, ATC '17, pp. 391–402 (2017)
43. Meng, L., Li, Y., Chen, L., Wang, Z., Wu, D., Zhou, Y., Xu, B.: Measuring discrimination to boost comparative testing for multiple deep learning models. In: 43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021, pp. 385–396. IEEE (2021)
44. Misiakos, P., Wendler, C., Püschel, M.: Learning dags from data with few root causes. In: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023 (2023)
45. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Discovery Science, pp. 264–269. Springer (2007)
46. Openja, M., Khomh, F., Foundjem, A., Jiang, Z.M.J., Abidi, M., Hassan, A.E.: An empirical study of testing machine learning in the wild. ACM Trans. Softw. Eng. Methodol. **34**(1), 7:1–7:63 (2025)
47. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019). DOI 10.1016/J.NEUNET.2019.01.012. URL `https://doi.org/10.1016/j.neunet.2019.01.012`
48. Poenaru-Olaru, L., Karpova, N., Cruz, L., Rellermeyer, J.S., van Deursen, A.: Is your anomaly detector ready for change? adapting aiops solutions to the real world. In: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI, pp. 222–233 (2024)
49. Prasad, P., Rich, C.: Market guide for AIOps platforms. Gartner Research (2018). Posted at `https://www.gartner.com/doc/3892967/market-guide-aiops-platforms`
50. Rajbahadur, G.K., Wang, S., Oliva, G.A., Kamei, Y., Hassan, A.E.: The impact of feature importance methods on the interpretation of defect classifiers. In: IEEE Transactions on Software Engineering (2021)
51. Rosà, A., Chen, L.Y., Binder, W.: Catching failures of failures at big-data clusters: A two-level neural network approach. In: Proceedings of the 2015 IEEE 23rd International Symposium on Quality of Service, IWQoS '15, pp. 231–236 (2015)
52. Rosà, A., Chen, L.Y., Binder, W.: Predicting and mitigating jobs failures in big data clusters. In: Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '15, pp. 221–230 (2015)
53. Scott, A.J., Knott, M.: A cluster analysis method for grouping means in the analysis of variance. Biometrics **30**(3), 507–512 (1974)
54. Song, L., Minku, L.L., Yao, X.: The impact of parameter tuning on software effort estimation using learning machines. In: Proceedings of the 9th International Conference on Predictive Models in Software Engineering, PROMISE '13 (2013)
55. Tantithamthavorn, C., Hassan, A.E.: An experience report on defect modelling in practice: Pitfalls and challenges. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '18, pp. 286–295 (2018)
56. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: Automated parameter optimization of classification techniques for defect prediction models. In: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, pp. 321–332 (2016)
57. Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K.: The impact of automated parameter optimization on defect prediction models. IEEE Transactions on Software Engineering **45**(7), 683–711 (2018)

58. Tsymbal, A.: The problem of concept drift: Definitions and related work. Computer Science Department, Trinity College Dublin **106**(2), 58 (2004)
59. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, pp. 226–235 (2003)
60. Wang, H., Yu, P.S., Han, J.: Mining Concept-Drifting Data Streams, pp. 789–802. Springer (2010)
61. Wang, L., Zhang, C., Ding, R., Xu, Y., Chen, Q., Zou, W., Chen, Q., Zhang, M., Gao, X., Fan, H., Rajmohan, S., Lin, Q., Zhang, D.: Root cause analysis for microservice systems via hierarchical reinforcement learning from human feedback. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023, pp. 5116–5125. ACM (2023)
62. Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., Ding, Y.: MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In: 19th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 22) (2022)
63. Wilkes, J.: Google cluster-usage traces v3. Technical report, Google Inc. (2020). Posted at `https://github.com/google/cluster-data/tree/master`
64. Wu, X., Li, H., Khomh, F.: On the effectiveness of log representation for log-based anomaly detection. Empir. Softw. Eng. **28**(6), 137 (2023)
65. Xu, Y., Sui, K., Yao, R., Zhang, H., Lin, Q., Dang, Y., Li, P., Jiang, K., Zhang, W., Lou, J.G., Chintalapati, M., Zhang, D.: Improving service availability of cloud systems by predicting disk error. In: Proceedings of the 2018 USENIX Annual Technical Conference, ATC '15, pp. 481–494 (2018)
66. Xue, J., Birke, R., Chen, L.Y., Smirni, E.: Managing data center tickets: Prediction and active sizing. In: Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN '16, pp. 335–346 (2016)
67. Xue, J., Birke, R., Chen, L.Y., Smirni, E.: Spatial-temporal prediction models for active ticket managing in data centers. IEEE Transactions on Network and Service Management **15**(1), 39–52 (2018)
68. Yang, F., Yin, W., Wang, L., Li, T., Zhao, P., Liu, B., Wang, P., Qiao, B., Liu, Y., Björkman, M., Rajmohan, S., Lin, Q., Zhang, D.: Diffusion-based time series data imputation for cloud failure prediction at microsoft 365. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023, pp. 2050–2055. ACM (2023)
69. Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X., Zhang, W.: Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021, pp. 1448–1460. IEEE (2021)
70. Yao, H., Tang, X., Wei, H., Zheng, G., Li, Z.: Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 5668–5675. AAAI Press (2019)
71. Zeng, Z., Zhang, Y., Xu, Y., Ma, M., Qiao, B., Zou, W., Chen, Q., Zhang, M., Zhang, X., Zhang, H., Gao, X., Fan, H., Rajmohan, S., Lin, Q., Zhang, D.: Traceark: Towards actionable performance anomaly alerting for online service systems. In: 45th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, SEIP@ICSE 2023, Melbourne, Australia, May 14-20, 2023, pp. 258–269. IEEE (2023)
72. Zhang, C., Peng, X., Sha, C., Zhang, K., Fu, Z., Wu, X., Lin, Q., Zhang, D.: Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning. In: 44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022, pp. 623–634. ACM (2022)
73. Zhang, J., Li, J., Yang, Z.: Dynamic robustness evaluation for automated model selection in operation. Inf. Softw. Technol. **178**, 107603 (2025)

74. Zhang, L., Jia, T., Jia, M., Yang, Y., Wu, Z., Li, Y.: A survey of AIOps for failure management in the era of large language models. arXiv preprint arXiv:2406.11213 (2024)
75. Zhang, S., Cohen, I., Goldszmidt, M., Symons, J., Fox, A.: Ensembles of models for automated diagnosis of system performance problems. In: 2005 International Conference on Dependable Systems and Networks (DSN 2005), 28 June - 1 July 2005, Yokohama, Japan, Proceedings (2005)
76. Zhao, N., Chen, J., Wang, Z., Peng, X., Wang, G., Wu, Y., Zhou, F., Feng, Z., Nie, X., Zhang, W., Sui, K., Pei, D.: 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, Virtual Event, USA, November 8-13, 2020, ESEC/FSE '20 (2020)