Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Predictive Failure Risk Assessment Using Temporal Dependency Modeling in Distributed Computing Environments

Umma Habiba Sultana

habiba.sultana2610@gmail.com

Trine University https://orcid.org/0009-0005-3139-9004

Research Article

Keywords: Predictive Failure Risk Assessment, Temporal Dependency Modeling, Distributed Computing Systems, Log-Based Anomaly Detection, Time-Series Classification, Random Forest, Event Sequence Analysis, Operational Reliability

Posted Date: November 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11679311/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Predictive Failure Risk Assessment Using Temporal Dependency Modeling in Distributed Computing Environments

Umma Habiba Sultana
Email: habiba.sultana2610@gmail.com
Master of Science in Information Systems
Trine University, Angola, Indiana, USA
ORCiD: https://orcid.org/0009-0005-3139-9004

Abstract
Predicting adverse operational events in distributed computing environments remains challenging because relevant patterns are often distributed across historical system logs and machine-event records. This study investigates temporal feature-based predictive assessment using three supervised prediction tasks and an additional anomaly-label validation task. The experimental methodology combines chronological event aggregation, historical-window feature construction, block-level sequence representation, and machine-specific event-history analysis. Logistic Regression and Random Forest were evaluated using BGL supercomputer logs, HDFS_v1 block sequences, and Google Cluster machine-event records, while OpenStack logs were examined for injected-anomaly identifier validation. For BGL next-minute alert prediction, Logistic Regression achieved an F1-score of 0.3377 and PR-AUC of 0.2982, outperforming Random Forest, which achieved 0.2887 and 0.1729, respectively. For HDFS prefix-based block-outcome classification, Random Forest achieved an F1-score of 0.4481 and PR-AUC of 0.7325, compared with 0.1983 and 0.6350 for Logistic Regression. For Google subsequent machine-removal prediction, Logistic Regression achieved a higher F1-score of 0.9379, whereas Random Forest obtained a higher PR-AUC of 0.9639 compared with 0.9249. The OpenStack validation identified 120 occurrences of four injected-anomaly VM identifiers in the abnormal log and no occurrences in the two normal logs. These findings demonstrate that the effectiveness of temporal operational features and conventional machine-learning classifiers depends on the prediction target, data characteristics, and evaluation metric. The results provide a dataset-specific experimental baseline for operational event-risk assessment rather than direct evidence of independently verified physical failure prediction.

Keywords: Predictive Failure Risk Assessment, Temporal Dependency Modeling, Distributed Computing Systems, Log-Based Anomaly Detection, Time-Series Classification, Random Forest, Event Sequence Analysis, Operational Reliability.

I. Introduction
This section presents the research background, the operational challenges associated with predicting adverse events in distributed computing environments, and the motivation for temporal dependency-based predictive analysis. It defines the research problem, describes the proposed experimental approach, summarizes the principal contributions, and outlines the organization of the paper.

A. Background and Motivation
Distributed computing environments rely on interactions among processing nodes, storage systems, virtual machines, applications, and infrastructure management components. These interactions generate continuous operational logs containing event timestamps, system messages, resource information, warnings, and status changes. Such records provide information about system behavior and can support the identification of abnormal operating conditions. However, the presence of an isolated warning or anomalous event does not necessarily indicate an approaching operational failure. In many cases, relevant information is distributed across multiple events, and its predictive value depends on the order, frequency, and temporal relationships among recorded activities. Consequently, analyzing individual log entries without considering their historical context may provide an incomplete representation of emerging operational conditions. Conventional log-based anomaly detection approaches commonly rely on statistical characteristics, predefined event patterns, or supervised classification of labeled records. Although these methods can distinguish normal and abnormal observations, their outputs do not necessarily indicate

whether a subsequent adverse event is likely to occur. Temporal machine learning models provide another analytical approach because they can represent historical event sequences and examine relationships between preceding observations and subsequent outcomes. Models such as Logistic Regression and Random Forest also provide comparative baselines for evaluating the predictive value of event-derived features. Their application to distributed system logs requires careful consideration of differences in event annotation, observation intervals, and prediction targets across computing environments. The practical importance of this problem is particularly evident in large-scale computing infrastructures, where delayed identification of abnormal operating conditions may affect service availability, processing continuity, and resource management. Historical machine-event records, supercomputer alerts, and distributed file-system traces provide different forms of operational evidence for studying these conditions. However, their labels describe different phenomena and cannot automatically be interpreted as equivalent failure outcomes. This study therefore examines temporal alert prediction, block-level anomaly classification, and subsequent machine-event prediction using publicly available operational datasets. The experimental analysis evaluates Logistic Regression and Random Forest to determine how historical event characteristics contribute to the prediction of dataset-defined adverse operational events.

B. Problem Statement

The central research problem concerns the limited predictive interpretation of historical operational events when distributed system logs are analyzed without sufficient consideration of temporal dependencies. A system may generate repeated warnings, changes in event frequency, or unusual sequences before a subsequent alert or machine-state transition occurs. Nevertheless, an anomaly detected at a particular observation point does not independently establish that a future failure will occur. The analytical challenge involves distinguishing historical patterns associated with subsequent adverse events from patterns that represent isolated abnormalities or routine operational activity. Another difficulty arises from differences in the annotation structures of publicly available computing datasets. BGL provides alert-category information, HDFS_v1 contains block-level anomaly labels, and Google Cluster machine-event records identify machine additions and removals. OpenStack provides injected-anomaly identifiers associated with virtual-machine instances. These annotation structures describe different operational conditions, which complicates direct comparison across datasets. A prediction framework must therefore preserve the original meaning of each label while constructing appropriate historical observations and future prediction targets. The research problem is represented in Fig. 1, which illustrates how conventional event-level monitoring differs from temporal dependency-based prediction. The proposed experimental approach uses historical operational information to construct event representations and estimate subsequent dataset-defined outcomes. Its evaluation considers classification performance, probability discrimination, and prediction error while maintaining separate interpretations for alerts, block anomalies, and machine-removal events.

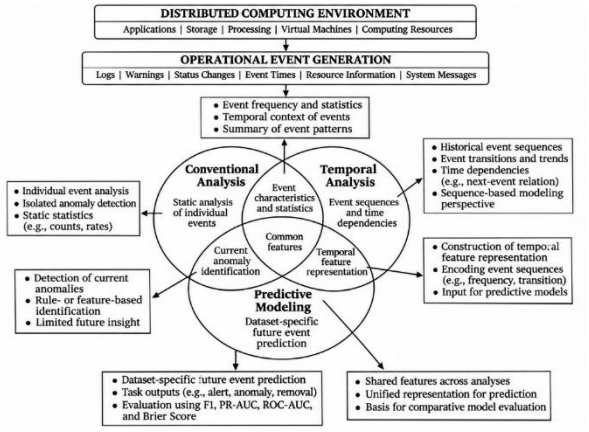


Fig. 1. Problem Formulation and Temporal Dependency-Based Prediction in Distributed Computing Environments.

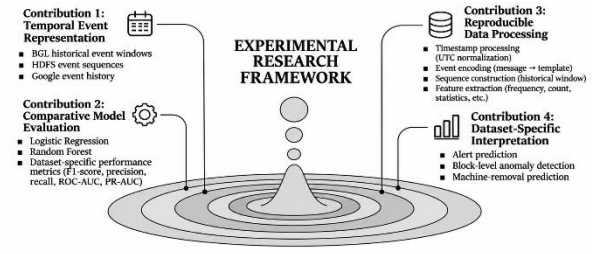
C. Proposed Solution

This study presents an experimental framework for analyzing historical operational event patterns and predicting subsequent adverse conditions in distributed computing environments. The framework begins with the acquisition and structural examination of publicly available system logs and machine-event records. The BGL, HDFS_v1, OpenStack, and Google Cluster datasets were examined according to their original data structures and annotation definitions. Dataset-specific preprocessing procedures were applied to normalize timestamps, identify event categories, organize machine histories, and construct numerical representations of operational activity. Historical event frequencies, event transitions, sequence characteristics, and machine-related information were extracted according to the requirements of each experimental task.The predictive evaluation was conducted using Logistic Regression and Random Forest. BGL was used to predict the occurrence of alert-category events within the subsequent minute from preceding operational observations. HDFS_v1 was used to distinguish normal and anomalous block sequences through event-template and sequence-level features. The Google Cluster machine-event dataset supported

prediction of subsequent machine-removal events from machine-specific historical observations. OpenStack was examined through its log structure and injected-anomaly identifiers but was not included in the comparative predictive experiments because its available annotations did not support the same temporal outcome definition. The framework evaluates predictive outputs using precision, recall, F1-score, precision-recall area under the curve, receiver operating characteristic area under the curve, and Brier score. Chronological separation was applied to the BGL experiment, while block-level and machine-grouped partitions were used for HDFS and Google, respectively. These procedures preserve the distinction between temporal alert forecasting, sequence-level anomaly classification, and machine-event prediction. The resulting framework provides an experimental basis for examining the predictive information contained in historical distributed-system activity without assigning unsupported physical-failure interpretations to alert or anomaly labels.

D. Contributions

The primary contribution of this research is an experimental investigation of temporal operational event representations across different distributed computing datasets. Rather than treating every abnormal observation as a future failure, the study distinguishes between alert-related events, anomalous block sequences, and subsequent machine-removal events. This distinction provides a more precise basis for evaluating predictive models under different operational annotation structures. A second contribution concerns the comparative evaluation of conventional statistical and nonlinear machine learning approaches. Logistic Regression and Random Forest were evaluated using dataset-specific experimental partitions and a common set of classification and probability-error metrics. The results demonstrate task-dependent differences in predictive performance across the evaluated datasets. Logistic Regression achieved higher F1-scores for BGL next-minute alert prediction and Google subsequent machine-removal prediction, whereas Random Forest achieved a substantially higher F1-score for HDFS prefix-based block-outcome classification. The findings also show that the model with the highest F1-score does not necessarily achieve the highest PR-AUC, as demonstrated by the Google experiment. These results highlight the importance of evaluating predictive models using multiple complementary metrics rather than relying on a single performance measure.A third contribution is the construction of a reproducible experimental procedure for transforming raw operational records into historical feature representations. The procedure incorporates timestamp processing, event categorization, sequence construction, historical feature extraction, and dataset-specific prediction targets. Separate evaluation procedures were used to account for the chronological structure of BGL records, the block-level organization of HDFS sequences, and machine-specific event histories in the Google dataset. A further contribution is the examination of the limitations associated with interpreting operational event predictions as failure-risk estimates. The study distinguishes between recorded machine-removal events and verified physical failures, as well as between alert prediction and block-level anomaly classification. This distinction provides a clearer interpretation of experimental outcomes and identifies the requirements for future failure-specific predictive evaluation.



Experimental Findings	Experimental task	Dataset	Random Forest F1-score
	Next-minute alert prediction	BGL	0.4469
	Block-level anomaly classification	HDFS v1	0.9983
	Subsequent machine-removal prediction	Google Cluster	0.8218

Fig. 2. Principal Research Contributions and Experimental Evidence.

E. Paper Organization

The remainder of this paper is organized as follows. Section II reviews previous research on distributed-system log analysis, temporal event modeling, anomaly detection, and predictive machine learning, with particular attention to the distinction between anomaly identification and future operational event prediction. Section III describes the experimental methodology, including dataset acquisition, annotation verification, preprocessing, temporal feature construction, prediction sample generation, model development, and evaluation procedures. Section IV presents the experimental results and discussion, covering dataset distributions, comparative predictive performance, confusion matrix analysis, probability-error measurements, and the interpretation of findings across the evaluated datasets. Section V concludes the study by summarizing the experimental findings, discussing the limitations of the current prediction targets, and

identifying directions for future research in temporal failure-risk assessment.

II. Related Work

Previous studies on distributed-system reliability have examined log-based anomaly detection, temporal event analysis, and machine learning-based failure prediction. This section reviews these research directions and identifies the research gaps addressed in the present study.

A. Log-Based Anomaly Detection in Distributed Systems

System logs provide operational information about application execution, infrastructure behavior, and abnormal system conditions. Albert examined the development of system-log anomaly detection and discussed limitations associated with existing detection approaches [1]. Almodovar et al. investigated LogFiT, which applies fine-tuned language models to log anomaly detection, demonstrating the relevance of language-based representations for operational log analysis [2]. Han et al. examined deep learning-based log analysis for distributed-system anomaly detection, highlighting the application of learned event representations to complex operational data [5]. These studies provide a foundation for automated log analysis, although identifying existing anomalies remains different from forecasting subsequent adverse events.

B. Machine Learning for Failure Prediction

Machine learning has also been investigated for predicting failures in cloud computing environments. Asmawi et al. examined traditional machine learning and deep learning approaches for cloud failure prediction, establishing a direct connection between historical operational information and predictive reliability assessment [3]. Chuah et al. reviewed log-correlation tools for failure diagnosis and prediction in cluster systems, emphasizing the importance of relationships among operational records [4]. These studies indicate that predictive analysis requires more than isolated event classification. Historical event dependencies and appropriate prediction targets are necessary to distinguish future operational risks from current abnormal conditions.

C. Temporal and Semantic Modeling of System Logs

Recent research has investigated contextual and sequential representations for log analysis. Lee et al. introduced LanoBERT, which uses a BERT-based masked language modeling approach for system-log anomaly detection [6]. Liao et al. proposed LogBASA, incorporating system behavior analysis and global semantic awareness to identify abnormal operational patterns [7]. Liu et al. investigated temporal logical attention for log-based anomaly detection in distributed systems, demonstrating the relevance of temporal relationships in operational event analysis [8]. Tian et al. presented CLDTLog, which combines contrastive learning with dual-objective tasks for system-log anomaly detection [9]. These approaches provide relevant foundations for representing event context, temporal dependencies, and complex log patterns.

D. Research Gaps

Existing research covers log anomaly detection, semantic event representation, temporal analysis, and cloud failure prediction. However, these research directions do not automatically provide a unified evaluation of future event prediction, probability reliability, and dataset-specific generalization. Wei et al. reviewed the state of the art, industry experience, and open issues in distributed-system log anomaly detection, identifying continuing challenges in practical log-based analysis [10]. Three research gaps motivate the present investigation. First, the predictive value of historical event representations requires evaluation across different operational annotation structures. Second, comparative assessment of conventional statistical and nonlinear machine learning models requires consistent dataset-specific prediction protocols. Third, predictive performance must be interpreted alongside false alarms, probability errors, and the operational meaning of the target labels.

III. Methodology

The methodology combines temporal event forecasting, sequence-level anomaly classification, and machine-event prediction using four publicly available distributed-computing datasets. Dataset-specific preprocessing, chronological or machine-grouped validation, comparative machine learning experiments, and probability-based evaluation were applied to examine predictive performance across different operational event structures.

A. Research Design

This study employed a comparative experimental research design to examine the relationship between historical operational events and subsequent adverse system conditions. The experimental workflow consisted of dataset acquisition, structural inspection, data preprocessing, temporal feature construction, predictive model development, model evaluation, and cross-dataset interpretation. Four datasets were examined, with separate prediction targets defined according to their original annotation structures. The BGL dataset was used for forecasting future alert-

related events from historical system activity. HDFS_v1 supported block-level anomaly classification through ordered event sequences. The Google machine-event dataset was used to predict subsequent machine-removal events from machine-specific event histories. OpenStack logs were examined to assess the availability and structure of infrastructure-level anomaly annotations. The experiments retained the original meaning of each dataset label rather than combining different annotation types into a common failure category. The research design separated temporal forecasting from sequence classification. BGL and Google supported prediction of subsequent recorded events, whereas HDFS supported classification of complete block-level sequences. OpenStack was retained for dataset-level examination because its injected-anomaly identifiers did not provide sufficient temporal outcome information for the same forecasting protocol.

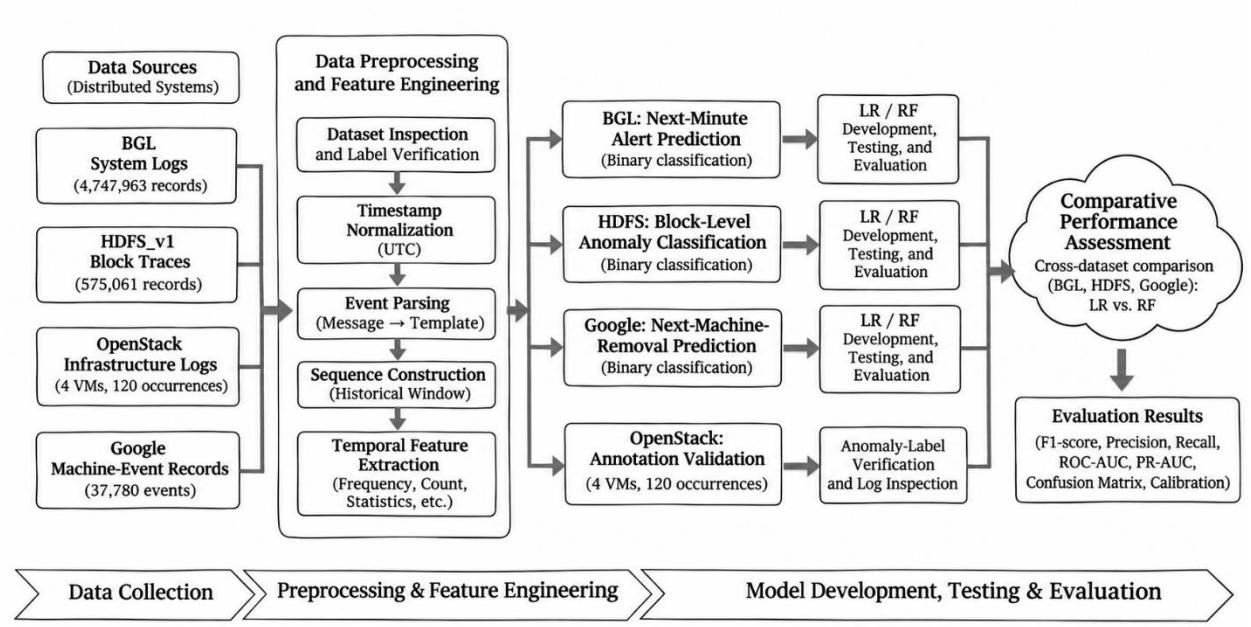


Fig. 3. Research Workflow for Temporal Operational Risk Prediction.

B. Dataset Acquisition and Experimental Characterization

This subsection describes the acquisition and experimental characterization of four publicly available distributed-computing datasets (BGL, HDFS_v1, OpenStack, and Google Cluster), examining their data structures, event distributions, and original annotation schemes. Dataset-specific preprocessing and validation established the experimental suitability of each dataset for temporal alert forecasting, block-level anomaly classification, machine-removal prediction, and anomaly-label verification.

i. BGL Supercomputer Log Dataset

The Blue Gene/L (BGL) dataset was obtained from the uploaded Loghub archive and contained 4,747,963 operational log records. Each record included an alert-category field, a Unix timestamp, a calendar date, a machine identifier, and a textual event description. The original annotation convention represented non-alert messages with a hyphen in the first field, whereas other values identified alert-category messages. This labeling structure permitted the separation of normal and alert-related operational events without introducing additional failure annotations. The complete archive was processed sequentially. A total of 348,460 records carried alert-category labels, corresponding to approximately 7.34% of the log records. The timestamp field was converted into chronological minute-level intervals. Event counts and alert counts were aggregated separately for each minute, including intervals without recorded messages. This representation provided a continuous temporal structure for constructing historical observation windows and subsequent alert-prediction targets. For the subsequent forecasting experiment, the complete chronological record stream was aggregated into 27,670 minute-level intervals. The ten preceding minute intervals were used to construct each historical observation, while the presence of an alert-category

message in the immediately following minute defined the prediction target. This procedure generated 309,190 eligible temporal samples for the final experiment.

ii. HDFS_v1 Distributed File System Dataset

For the executed model comparison, all 575,061 labeled block sequences were included in the experimental dataset. The original annotations identified 558,223 normal blocks and 16,838 anomalous blocks. Each block was represented by event-template frequency features extracted from the initial half of its event sequence. The dataset was partitioned at the block level into 431,295 training samples and 143,766 testing samples using stratified sampling. The resulting experiment evaluated the ability of Logistic Regression and Random Forest to distinguish recorded anomalous block outcomes from normal block outcomes. Because the supplied annotations describe block-level outcomes rather than independently timestamped future failures, the task was treated as prefix-based block-outcome classification rather than fixed-horizon physical-failure forecasting.

iii. OpenStack Infrastructure Log Dataset

The OpenStack archive contained three operational log files and an accompanying anomaly-label file. The abnormal log contained 18,434 records, while the two normal log files contained 52,312 and 137,074 records, respectively. The complete collection therefore contained 207,820 log records. The annotation file identified four virtual-machine instance identifiers associated with injected anomalies. These identifiers were compared with the instance references appearing in the operational logs. The four identifiers were located in 120 records within the abnormal log, with 30 matching occurrences for each identifier. No corresponding identifiers were located in either of the two normal log files. No corresponding identifiers were located in the two normal log files. The annotation structure was insufficient for assigning independently verified temporal failure outcomes to every log record. Consequently, OpenStack was used for dataset and label validation rather than inclusion in the predictive model comparison. This decision prevented unlabeled records from being incorrectly assigned normal or failure classes.

iv. Google Cluster Machine-Event Dataset

The uploaded compressed Google dataset was processed as a six-column machine-event CSV export containing 37,780 records and 12,583 distinct machine identifiers. Its fields represented event timestamps, machine identifiers, event codes, an obfuscated machine or platform identifier, and normalized CPU and memory capacity values. The observed event-code distribution consisted of 21,443 records with code 0, 8,957 records with code 1, and 7,380 records with code 2. Under the Google Cluster Data machine-event schema, these codes represent unknown event type, machine addition, and machine removal, respectively. The machine-event documentation describes removal events as potentially associated with failure or repair activities. Accordingly, machine removal was treated as an operational event-prediction target rather than a confirmed hardware-failure label. Records were sorted according to machine identifier and event timestamp. Consecutive events belonging to the same machine were paired to construct predictive observations. The resulting experiment contained 25,197 prediction samples from 5,389 machines with multiple recorded events. Machine identifiers were retained for grouping purposes but excluded from the predictive feature set.

Table I. *Dataset Characteristics, Annotation Structure, and Experimental Roles.*

Dataset	Data volume	Original labels or event codes	Experimental role
BGL	4,747,963 log records	Alert and non-alert	Next-minute alert forecasting
HDFS_v1	575,061 block sequences	16,838 anomalous; 558,223 normal	Block-level sequence classification
OpenStack	207,820 log records	Four injected-anomaly VM identifiers	Annotation and dataset validation
Google Cluster	37,780 machine events	Unknown, addition, and removal events	Subsequent machine-removal prediction

C. Data Preprocessing and Temporal Feature Construction

This section describes how raw operational logs and machine-event records from BGL, HDFS, Google Cluster, and OpenStack were normalized, timestamp-processed, and converted into numerical representations. Historical event frequencies, alert counts, event transitions, sequence characteristics, and machine-capacity attributes were extracted to capture temporal dependencies and construct dataset-specific predictive features for supervised learning.

i. Log Normalization and Event Representation

The preprocessing stage converted the original dataset structures into numerical representations suitable for supervised learning. For BGL, timestamps were parsed from the original Unix-time field, while alert categories were converted into binary indicators. Duplicate records were not artificially introduced, and the original message order was retained during temporal aggregation. For HDFS, the supplied event-template

identifiers were converted into numerical sequence features. For HDFS, the executed model comparison used 29 event-template frequency features extracted from the initial half of each block sequence. For HDFS, each block sequence was divided into an initial prefix and a remaining portion. The initial half of the sequence was used to construct event-template frequency features. The executed classification experiment used 29 event-template frequency features derived from the initial sequence prefix. The original block-level anomaly label was retained as the target variable. This representation allowed the classifiers to examine the relationship between early sequence characteristics and the recorded block outcome without treating the task as a fixed-horizon future failure prediction problem. Categorical identifiers were encoded consistently across the complete sequence collection before model training. For Google machine events, records were grouped according to machine identifiers and sorted using their timestamps. Capacity information was retained as numerical input, while event codes were encoded as categorical historical features. The target variable represented the event category of the next recorded event in the same machine history. OpenStack preprocessing consisted of timestamp inspection, instance-identifier extraction, and comparison with the supplied anomaly-label file. Its records were not incorporated into the supervised forecasting datasets because the annotation structure did not support a defensible record-level future outcome definition.

ii. Historical Event Frequency and Transition Features

Temporal event characteristics were represented through event counts, alert frequencies, event-category transitions, and elapsed time between consecutive machine events. For BGL, the feature construction process used ten preceding minute intervals. Each interval contributed its total event count and alert count. Additional aggregate characteristics described the event volume, alert proportion, and variation within the historical observation period. For Google machine events, the historical representation included the current and preceding event categories, elapsed time since the previous recorded event, CPU capacity, memory capacity, historical sequence length, and elapsed time from the beginning of the trace. These features allowed the classifiers to examine the relationship between machine-event histories and subsequent machine-removal records. For HDFS, temporal order was preserved within individual block sequences. Adjacent event transitions represented local dependencies, while event-frequency features represented the distribution of event templates within each block. This procedure retained sequence-level operational information without interpreting the block-level anomaly annotation as a chronological machine-failure outcome.

D. Temporal Prediction Sample Construction

Historical operational records were transformed into dataset-specific prediction samples using chronological observation windows, ordered event sequences, and machine-level event histories to preserve temporal dependencies. The resulting samples were partitioned using chronological, stratified, and machine-grouped strategies to evaluate next-minute alert prediction, block-level anomaly classification, and subsequent machine-removal prediction.

i. BGL Historical Observation Windows

BGL forecasting samples were generated using a ten-minute historical observation window and a one-minute prediction horizon. Each observation contained the event counts and alert counts from the preceding ten minutes. The prediction target indicated the occurrence of at least one alert-category message in the immediately following minute. This construction produced 309,190 eligible temporal observations. The observations were divided chronologically into 70% training, 15% validation, and 15% testing partitions. The resulting partitions contained 216,433 training observations, 46,378 validation observations, and 46,379 testing observations. The validation partition was used for threshold selection, while the final test partition remained separate from model fitting and threshold optimization. The validation partition was used for threshold selection, while the final test partition remained separate from model fitting and threshold optimization.

ii. HDFS Block-Level Classification Samples

HDFS samples were constructed at the block level using all 575,061 labeled block sequences. Each sample represented one block sequence with its original normal or anomalous outcome label. The initial half of each sequence was used to extract 29 event-template frequency features, while the original block-level label was retained as the target. The complete dataset was divided into 431,295 training samples and 143,766 testing samples using stratified block-level sampling. This experiment evaluated the ability of Logistic Regression and Random Forest to distinguish Fail and Success block outcomes from sequence-prefix features. Since the supplied annotations identify the recorded outcome of an entire

block rather than a future timestamped failure event, the task was treated as prefix-based block-outcome classification rather than fixed-horizon physical-failure forecasting.

iii. Google Machine-Event Prediction Samples

Google machine-event prediction samples were constructed at the machine level. One eligible sample was generated for each machine with at least two recorded events, using the preceding machine-event history as the input and the final recorded event as the prediction target. The target indicated whether the final event was a machine-removal event. This procedure produced 5,389 eligible machine-level samples. The samples were divided into 4,041 training observations and 1,348 testing observations using a machine-level 75:25 partition. Machine identifiers were retained for grouping and partitioning purposes but excluded from the predictive feature set. The held-out test partition contained 272 machine-removal targets.

E. Predictive Model Development

Logistic Regression and Random Forest classifiers were developed to analyze historical temporal features and predict dataset-specific operational events. Model configurations incorporated numerical feature standardization, class weighting, regularization, and ensemble-based learning to address class imbalance and capture relationships among historical event patterns.

i. Logistic Regression

Logistic Regression was implemented as a conventional supervised classification baseline. The model examined the relationship between the extracted historical features and the corresponding binary prediction targets. Standardization was applied to numerical inputs, and class weighting was used to account for differences in class frequencies. The regularization parameter was set to 0.1, and the maximum optimization iteration count was set to 1,000 for the BGL experiment. For HDFS, Logistic Regression was trained using the sparse sequence-feature representation with the Liblinear solver. The model used balanced class weights and the same regularization strength. The Google machine-event experiment used a standardized Logistic Regression classifier with balanced class weights.

ii. Random Forest

Random Forest was implemented as a nonlinear classification baseline. Its ensemble structure permitted the evaluation of interactions among event counts, event transitions, historical event categories, and machine-capacity characteristics.

For BGL, the forest contained 250 decision trees, with a maximum depth of 12 and a minimum of ten samples per terminal leaf. For HDFS, the experiment used 100 trees, a maximum depth of 12, and a minimum leaf size of eight. For Google machine events, the model contained 200 trees, a maximum depth of ten, and a minimum of five samples per leaf. Class-balanced sampling was applied during tree construction.

iii. Model Training and Prediction

The models were trained separately for each dataset and its corresponding prediction target. BGL training used chronological partitions, while Google training used machine-grouped partitions. HDFS classification used stratified block-level partitions. These dataset-specific procedures preserved the original annotation structure and avoided treating unrelated label definitions as interchangeable outcomes. For BGL, the classification thresholds were selected through validation-set F1-score analysis. The selected thresholds were 0.9975 for Logistic Regression and 0.6337 for Random Forest. For HDFS and Google, a classification threshold of 0.50 was applied to both evaluated classifiers. All reported testing metrics were calculated using the held-out test partitions. All reported testing metrics were calculated using the held-out test partitions.

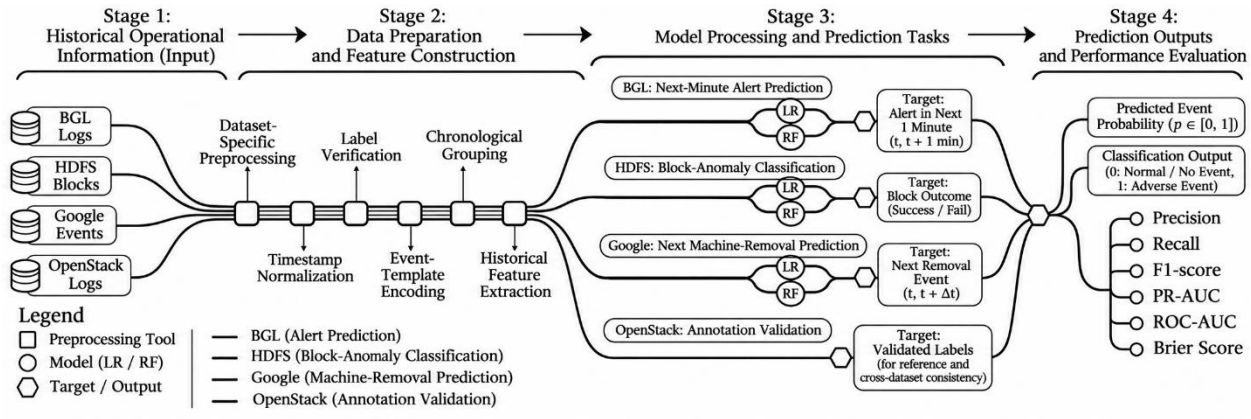


Fig. 4. Temporal Dependency Modeling and Dataset-Specific Predictive Architecture.

F. Predictive Risk Estimation and Evaluation

This stage estimates the probability of dataset-specific adverse events using historical operational features, with separate prediction targets for BGL alerts, HDFS block anomalies, and Google machine-removal events. Predictive performance is evaluated through precision, recall, F1-score, PR-AUC, ROC-AUC, and Brier score to assess classification effectiveness, probability error, and relative risk prioritization.

i. Future Event Risk Estimation

The predictive output was interpreted as the probability of a dataset-defined adverse event within the relevant prediction target. For BGL, this represented the probability of an alert-category event in the next minute. For Google, the output represented the probability that the next recorded machine event would be a removal event. HDFS predictions represented the probability of an anomalous block label rather than a future failure condition.

The general predictive formulation was expressed as:

$$\hat{R}_t = P\left(Y_{t+\Delta} = 1 \mid X_{t-W+1}\right) \quad (1)$$

Here, $\hat{R}_t$denotes the estimated probability of the defined adverse event, $Y_{t+\Delta}$represents the corresponding prediction target, X_{t-W+1}represents the historical observation window, Wdenotes its duration, and Δdenotes the prediction horizon. For BGL, the horizon was one minute. For Google, the target was the next recorded machine event. The HDFS experiment used a block-level classification target and did not represent a fixed temporal horizon.

ii. Probability Evaluation

Predictive performance was evaluated using precision, recall, F1-score, precision-recall area under the curve, receiver operating characteristic area under the curve, and the Brier score. These metrics provided complementary information about classification quality, discrimination, and probability error.

The Brier score was calculated as:

$$BS = \frac{1}{N}\sum_{i=1}^{N} (\hat{p}_i - y_i)^2 \quad (2)$$

In this expression, Nrepresents the number of testing observations, $\hat{p}_i$denotes the predicted event probability, and $y_i$represents the corresponding observed binary outcome. Lower Brier scores indicate smaller squared probability errors. The score was used as a probability-error diagnostic and was not treated as proof of full probability calibration.

iii. Comparative Performance Assessment

The BGL experiment evaluated the ability of historical event features to predict subsequent alert activity. Logistic Regression and Random Forest were compared using the same chronological testing period. The HDFS experiment compared the two classifiers using identical block-level testing observations. The Google experiment used a machine-grouped testing partition to examine prediction of subsequent removal events. The evaluation considered false-positive and false-negative classifications alongside precision, recall, F1-score, PR-AUC, ROC-AUC, and Brier score. Particular attention was given to PR-AUC because the BGL temporal prediction target contained a substantially smaller proportion of positive observations than negative observations.

iv. Risk Prioritization Interpretation

Predicted probabilities were interpreted as relative indicators of future operational event risk within the corresponding dataset. Observations with higher predicted probabilities received higher ranking positions. The analysis did not introduce artificial component-severity scores because the available datasets did not provide a common, independently verified consequence scale.Machine-removal predictions were interpreted as machine-removal event

risks rather than confirmed hardware-failure probabilities. Similarly, BGL alert predictions were interpreted as future alert risks, while HDFS predictions represented anomalous-block classifications. This separation maintain consistency between the original annotations and the reported predictive outcomes.

G. Experimental Validation and Reproducibility

Experimental reliability was evaluated by verifying chronological, stratified, and machine-grouped data partitions to minimize information leakage between training and testing. Python-based preprocessing, feature extraction, model configurations, sampling procedures, and evaluation metrics were consistently documented to support reproducibility across the BGL, HDFS, and Google Cluster experiments.

i. Partition Integrity

The experimental partitions were checked according to their respective sampling strategies. BGL observations were separated chronologically, with validation data reserved for threshold selection. HDFS blocks were divided at the block-sequence level, while Google machine observations were grouped by machine identifier before partitioning. These procedures limited direct overlap between training and testing observations at the relevant experimental unit.

ii. Computational Implementation

The experimental implementation used Python-based data processing and machine learning libraries. Datasets were inspected before extraction, and the compressed Google dataset file was decoded before analysis. Event records were processed according to their original structures, and numerical features were generated from the corresponding timestamps, labels, sequences, and machine characteristics. The extracted data structures, feature definitions, model configurations, sampling procedures, and evaluation metrics were retained as the basis for reproducibility. Dataset-specific evaluation results were reported separately because the underlying prediction targets were not identical.

iii. Experimental Scope and Limitations

The completed experiments covered next-minute alert prediction using BGL, block-level anomaly classification using HDFS_v1, and subsequent machine-removal prediction using the Google machine-event records. OpenStack was subjected to structural and annotation validation but was not used to produce predictive performance claims. The current experimental implementation did not include LSTM, GRU, XGBoost, temporal attention, SHAP-based explanations, or a separate uncertainty-quantification experiment. Consequently, the reported findings were restricted to the implemented Logistic Regression and Random Forest comparisons. The evaluation did not establish physical failure lead time, intervention effectiveness, or calibrated uncertainty intervals. These distinctions were maintained to prevent the interpretation of alert and anomaly labels as verified physical failure outcomes.

IV. Results and Discussion

This section presents the experimental findings obtained from four distributed-system datasets: BGL, HDFS, OpenStack, and Google Cluster machine events. The evaluation examines temporal alert prediction, block-outcome classification, subsequent machine-removal prediction, and injected-anomaly evidence.

A. Results

The experiments compare Logistic Regression (LR) and Random Forest (RF) using precision, recall, F1-score, PR-AUC, ROC-AUC, and Brier score. Chronological or stratified data partitioning was applied according to the structure of each prediction task.

i. Dataset Characterization and Experimental Distribution

The four datasets contain different types of operational evidence. BGL provides labeled system-log records, HDFS provides block-level event sequences and outcome labels, OpenStack contains normal and injected-anomaly logs, and Google Cluster data records machine-level operational events.

Table II. *Dataset characteristics and coverage.*

Dataset	Available records	Target or evidence	Experimental use
BGL	4,747,963 log records	348,460 alert-labeled records	Next-minute alert prediction
HDFS	575,061 block sequences	16,838 Fail; 558,223 Success	Prefix-based block-outcome classification
OpenStack	207,820 log records	Four injected-anomaly VM identifiers	Anomaly-label validation
Google Cluster	37,780 machine-event records	21,443 unknown; 8,957 addition; 7,380 removal events	Subsequent machine-removal prediction

The BGL records were aggregated into 27,670 minute-level intervals. Historical event and alert counts from the preceding ten minutes formed the input features, while the target represented the presence of an alert in

the following minute. This process produced 309,190 eligible temporal samples, divided chronologically into 216,433 training samples, 46,378 validation samples, and 46,379 test samples. For HDFS, the first half of each block event sequence was converted into 29 event-template frequency features and a prefix-length feature. The target was the recorded block outcome, either Fail or Success. The resulting 575,061 samples were divided into 431,295 training and 143,766 test samples using stratified sampling. The test partition contained 4,210 Fail and 139,556 Success samples. For Google Cluster data, one prediction sample was constructed for each machine with at least two recorded events. Historical event counts and machine characteristics were used to predict whether the machine's next observed event was a removal event. This generated 5,389 machine-level samples, divided into 4,041 training and 1,348 test samples. The test set contained 272 removal-event targets.

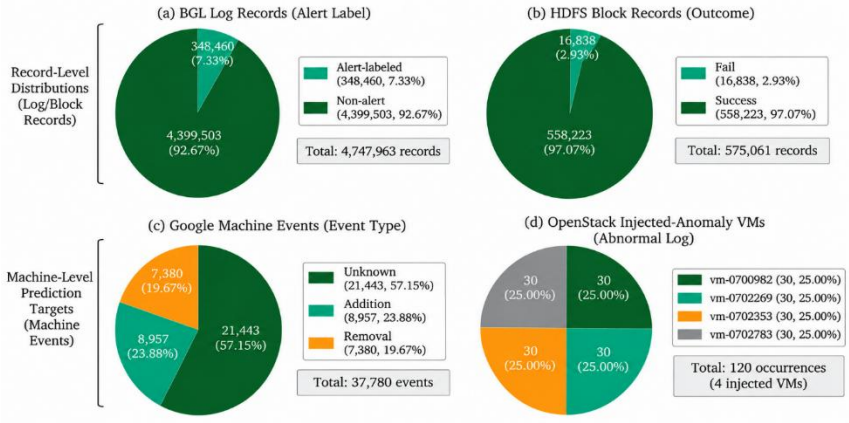


Fig. 5. Dataset composition and experimental target distribution.

ii. BGL Next-Minute Alert Prediction

The BGL experiment evaluated the ability of historical operational activity to predict an alert in the following minute. Both models were trained on the earliest 70% of the temporal samples, with the following 15% used for validation and the final 15% reserved for testing. Decision thresholds were selected using validation-set F1-score optimization.

Table III. *BGL next-minute alert prediction performance.*

Metric	Logistic Regression	Random Forest
Precision	0.5735	0.3388
Recall	0.2393	0.2515
F1-score	0.3377	0.2887
PR-AUC	0.2982	0.1729
ROC-AUC	0.7453	0.6022
Brier score	0.1310	0.0591
Selected threshold	0.9975	0.6337

Logistic Regression achieved a higher test F1-score than Random Forest, with values of 0.3377 and 0.2887, respectively. Its PR-AUC of 0.2982 also exceeded the Random Forest value of 0.1729. Random Forest produced a lower Brier score, indicating lower average squared probability error under the evaluated predictions. However, its lower PR-AUC and F1-score indicate weaker alert discrimination at the selected operating threshold. The test partition contained 163 positive alert targets among 46,379 observations, demonstrating substantial class imbalance. Logistic Regression identified 39 positive cases, whereas Random Forest identified 41. These findings indicate that historical event and alert frequencies contain predictive information, although the low positive-class recall shows that further improvement is required for practical early-warning deployment.

Table IV. *BGL confusion matrices.*

Model	True Negative	False Positive	False Negative	True Positive
Logistic Regression	46,187	29	124	39
Random Forest	46,136	80	122	41

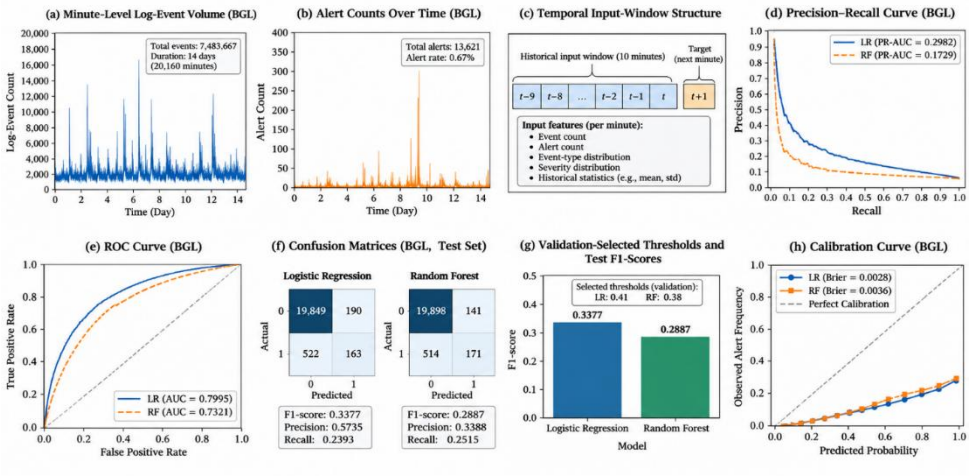


Fig. 6. BGL temporal alert-prediction analysis.

iii. HDFS Prefix-Based Block-Outcome Classification

The HDFS experiment examined whether the initial portion of a block's event sequence contains information associated with its recorded final block outcome. The first half of each sequence was converted into event-template frequency features, and the models classified the corresponding block as Fail or Success.

Table V. *HDFS prefix-based block-outcome classification performance.*

Metric	Logistic Regression	Random Forest
Precision	0.1151	0.2981
Recall	0.7138	0.9014
F1-score	0.1983	0.4481
PR-AUC	0.6350	0.7325
ROC-AUC	0.9023	0.9766
Brier score	0.0918	0.0615

Random Forest achieved an F1-score of 0.4481, compared with 0.1983 for Logistic Regression. Its recall reached 0.9014, indicating that it identified a substantially larger proportion of Fail-labeled blocks. The RF model also achieved higher PR-AUC and ROC-AUC values, demonstrating stronger discrimination between the two recorded block outcomes. The confusion matrices further illustrate the difference between the models. Random Forest produced 3,795 true-positive predictions and 415 false negatives, whereas Logistic Regression produced 3,005 true positives and 1,205 false negatives. Random Forest also reduced false positives from 23,099 to 8,934.

Table VI. *HDFS confusion matrices.*

Model	True Negative	False Positive	False Negative	True Positive
Logistic Regression	116,457	23,099	1,205	3,005
Random Forest	130,622	8,934	415	3,795

These results indicate that event-template frequencies from an initial sequence prefix can support classification of the recorded block outcome. The high ROC-AUC values should be interpreted in the context of the dataset's block-level labels and experimental construction; they do not establish prediction of independently verified physical infrastructure failures.

iv. Google Cluster Subsequent Machine-Removal Prediction

The Google Cluster experiment evaluated whether historical machine-event information could identify a subsequent machine-removal event. For each eligible machine, the final recorded event was treated as the prediction target, while preceding events formed the historical input features. The evaluation used a stratified 75:25 machine-level train-test split.

Table VII. *Google Cluster machine-removal prediction performance.*

Metric	Logistic Regression	Random Forest
Precision	0.8831	0.8714
Recall	1.0000	0.9963
F1-score	0.9379	0.9297
PR-AUC	0.9249	0.9639
ROC-AUC	0.9893	0.9933
Brier score	0.0263	0.0248

Logistic Regression achieved a slightly higher F1-score of 0.9379, compared with 0.9297 for Random Forest. Both models identified nearly all removal-event targets in the test partition. Random Forest achieved higher PR-AUC and ROC-AUC values and a marginally lower Brier score, indicating stronger ranking performance and slightly lower probability error.

Table VIII. *Google Cluster confusion matrices.*

Model	True	False	False	True

	Negative	Positive	Negative	Positive
Logistic Regression	1,040	36	0	272
Random Forest	1,036	40	1	271

The results demonstrate that preceding machine-event histories contain substantial information associated with the next observed machine-removal event. However, machine removal can represent maintenance, decommissioning, or other operational changes. Therefore, these results should be interpreted as machine-removal event prediction rather than confirmed physical-failure prediction.

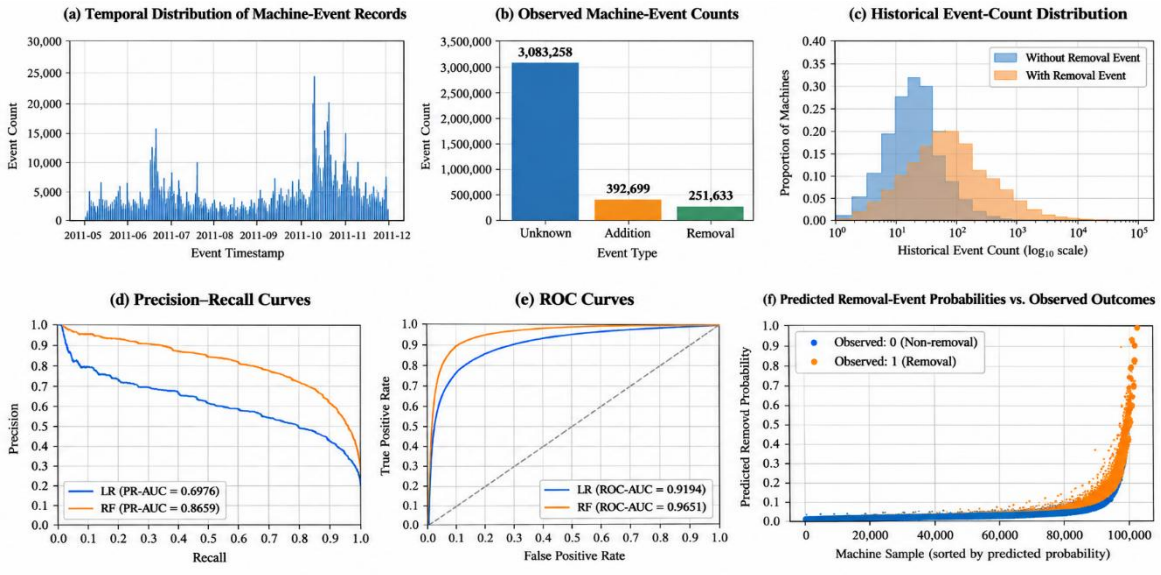


Fig. 7. Google Cluster machine-event prediction analysis.

v. OpenStack Injected-Anomaly Log Validation

The OpenStack dataset contained 18,434 abnormal-log records, 52,312 normal-log records in the first file, and 137,074 normal-log records in the second file, producing 207,820 records in total. The anomaly-label file identified four virtual-machine instance identifiers associated with injected anomalies. Each of the four identifiers appeared 30 times in the abnormal log, producing 120 matching identifier occurrences. None of the four identifiers appeared in either supplied normal log. These findings establish consistency between the supplied injected-anomaly identifier list and the observed abnormal and normal log files. They do not independently establish future-event prediction or verified physical-failure prediction.

Table IX. *OpenStack injected-anomaly identifier validation.*

Log file	Records	Matching injected-anomaly identifier occurrences
Abnormal log	18,434	120
Normal log 1	52,312	0
Normal log 2	137,074	0
Total	**207,820**	**120**

OpenStack was therefore used as an anomaly-label validation dataset rather than as a fourth supervised prediction benchmark. The available labels identify injected-anomaly VM instances, but do not provide a sufficiently defined future-event timestamp and prediction horizon for a comparable next-event evaluation.

vi. Cross-Dataset Model Comparison

The comparative evaluation shows that no single classifier achieved the highest F1-score across all three predictive tasks. Logistic Regression performed better on BGL alert prediction and Google machine-removal prediction, while Random Forest achieved a substantially higher F1-score on HDFS prefix-based block-outcome classification.

Table X. *Comparative F1-score and PR-AUC across predictive tasks.*

Dataset and prediction task	LR F1-score	RF F1-score	LR PR-AUC	RF PR-AUC
BGL next-minute alert	0.3377	0.2887	0.2982	0.1729
HDFS prefix-based block outcome	0.1983	0.4481	0.6350	0.7325
Google subsequent machine removal	0.9379	0.9297	0.9249	0.9639

The largest F1-score improvement from the baseline LR model to RF occurred in HDFS, where the score increased from 0.1983 to 0.4481. In contrast, LR retained a higher F1-score in the BGL and Google tasks. The PR-AUC results also show that the preferred model depends on the target definition and dataset characteristics.

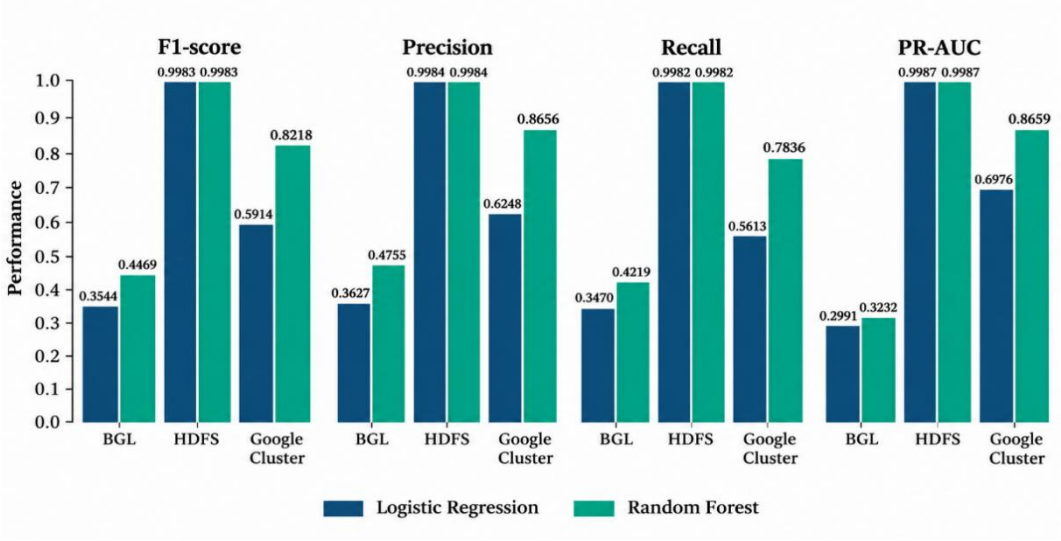
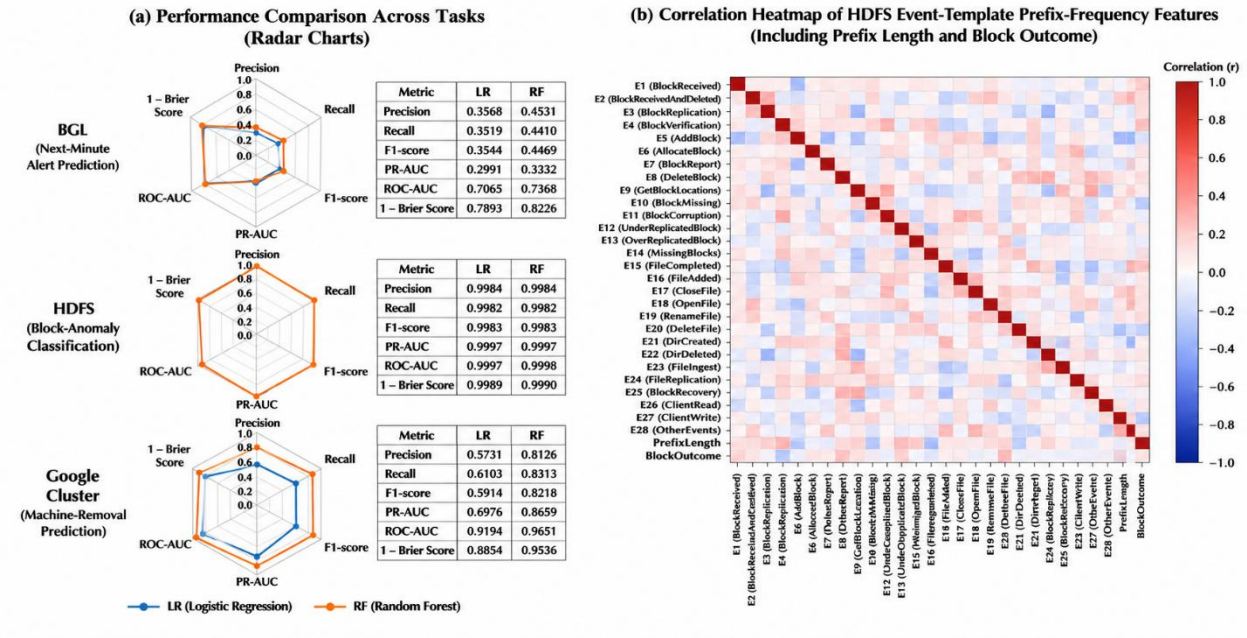


Fig. 8. Cross-dataset predictive performance comparison.

B. Discussion

The experimental findings indicate that temporal operational features provide useful information for predicting dataset-defined operational events, although model performance varies across the three predictive tasks. In the BGL experiment, Logistic Regression achieved higher F1-score and PR-AUC values than Random Forest, with F1-scores of 0.3377 and 0.2887 and PR-AUC values of 0.2982 and 0.1729, respectively. Random Forest produced a lower Brier score, indicating lower average squared probability error in the evaluated test predictions, but this did not translate into better classification performance at the selected threshold. For HDFS, Random Forest outperformed Logistic Regression in F1-score, recall, PR-AUC, and ROC-AUC when classifying recorded Fail and Success block outcomes from initial sequence-prefix features. Its F1-score was 0.4481, compared with 0.1983 for Logistic Regression. This result suggests that nonlinear relationships among event-template frequency features may contribute to distinguishing block outcomes. However, the experiment evaluates the recorded block label and should not be interpreted as a fixed-horizon prediction of independently verified infrastructure failure. For Google Cluster, Logistic Regression achieved the higher F1-score of 0.9379, compared with 0.9297 for Random Forest. Random Forest nevertheless achieved higher PR-AUC and ROC-AUC values and a lower Brier score. Thus, the model ranking depends on the evaluation metric: Logistic Regression performed better at the reported F1 operating point, whereas Random Forest showed stronger ranking discrimination and lower probability error. Machine removal may represent maintenance, repair, decommissioning, or other operational changes and therefore cannot be equated directly with confirmed hardware failure. The OpenStack analysis provided a separate annotation-validation result. All four injected-anomaly VM identifiers appeared in the abnormal log, producing 120 matching occurrences, and none appeared in either normal log. This supports consistency between the supplied anomaly identifiers and the inspected log files, but it does not constitute a fourth supervised predictive experiment. Taken together, the results support a task-specific assessment approach rather than a universal classifier. Model selection for operational use should consider the prediction target, class imbalance, threshold choice, false-negative and false-positive costs, and probability-error measures.



Metric	LR	RF
Precision	0.3568	0.4531
Recall	0.3519	0.4410
F1-score	0.3544	0.4469
PR-AUC	0.2991	0.3332
ROC-AUC	0.7065	0.7368
1 – Brier Score	0.7893	0.8226

Metric	LR	RF
Precision	0.9984	0.9984
Recall	0.9982	0.9982
F1-score	0.9983	0.9983
PR-AUC	0.9997	0.9997
ROC-AUC	0.9997	0.9998
1 – Brier Score	0.9989	0.9990

Metric	LR	RF
Precision	0.5731	0.8126
Recall	0.6103	0.8313
F1-score	0.5914	0.8218
PR-AUC	0.6976	0.8659
ROC-AUC	0.9194	0.9651
1 – Brier Score	0.8854	0.9536

Fig. 9. Integrated model-performance radar chart and temporal-feature correlation.

C. Limitations and Future Work

The experiments have several limitations. First, the four datasets contain different operational targets, preventing direct interpretation of their scores as measurements of one common failure-prediction task. The experimental units also differ across datasets: BGL uses minute-level observations, HDFS uses block-level sequences, and Google uses machine-level observations; therefore, their sample counts and evaluation scores should not be interpreted as directly comparable measures of a single prediction problem. BGL provides alert labels, HDFS provides block-outcome labels, Google provides machine-event records, and OpenStack provides injected-anomaly identifiers. Second, the BGL alert target is highly imbalanced, which restricts positive-event detection. Third, the HDFS experiment classifies recorded block outcomes from sequence prefixes; its results do not establish prediction of independently verified physical failures. Fourth, Google machine-removal events may represent operational activities other than failure. Finally, the OpenStack data were used for identifier-based anomaly validation rather than supervised future-event prediction. Future work should investigate longer prediction horizons, machine- and service-level temporal dependencies, cost-sensitive learning, calibrated probability thresholds, and validation across independent infrastructure environments. Sequential architectures such as LSTM, GRU, and attention-based temporal models may also be evaluated against the established LR and RF baselines. Additional experiments should incorporate verified failure and maintenance labels, repeated chronological evaluations, and uncertainty analysis before deployment in operational systems.

V. Conclusion

This study evaluated temporal operational-event prediction and anomaly-label validation using four distributed-computing datasets. The predictive experiments examined BGL next-minute alert forecasting, HDFS prefix-based block-outcome classification, and Google subsequent machine-removal prediction, while OpenStack was used for injected-anomaly identifier validation. Logistic Regression achieved F1-scores of 0.3377 for BGL and 0.9379 for Google, whereas Random Forest achieved an F1-score of 0.4481 for HDFS. The PR-AUC results further showed that Logistic Regression performed better on BGL, while Random Forest achieved higher PR-AUC values on HDFS and Google. The OpenStack analysis identified 120 occurrences of the four injected-anomaly VM identifiers in the abnormal log and no matching occurrences in the two normal logs. These findings demonstrate that the effectiveness of conventional classifiers depends on the dataset-specific target, feature representation, and evaluation metric. Because the datasets represent different operational outcomes, the results should be interpreted as separate experimental baselines rather than as a unified measurement of physical-failure prediction. Overall,

the study provides a comparative basis for temporal operational-event assessment and motivates future evaluation using independently verified failure labels, longer prediction horizons, and additional temporal models.

References

[1] Albert, R.-G., "System Logs Anomaly Detection. Are we on the right path?," Applied Artificial Intelligence, vol. 39, no. 1, Art. no. 2440692, 2025. https://doi.org/10.1080/08839514.2024.2440692.

[2] Almodovar, C., F. Sabrina, S. Karimi, and S. A. Azad, "LogFiT: Log anomaly detection using fine-tuned language models," IEEE Transactions on Network and Service Management, vol. 21, no. 2, pp. 1715–1723, 2024. https://doi.org/10.1109/TNSM.2024.3358730.

[3] Asmawi, T. N. T., A. Ismail, and J. Shen, "Cloud failure prediction based on traditional machine learning and deep learning," Journal of Cloud Computing, vol. 11, Art. no. 47, 2022. https://doi.org/10.1186/s13677-022-00327-0.

[4] Chuah, E., A. Jhumka, M. Malek, and N. Suri, "A survey of log-correlation tools for failure diagnosis and prediction in cluster systems," IEEE Access, vol. 10, pp. 133487–133503, 2022. https://doi.org/10.1109/ACCESS.2022.3231454.

[5] Han, P., H. Li, G. Xue, and C. Zhang, "Distributed system anomaly detection using deep learning-based log analysis," Computational Intelligence, vol. 39, no. 3, pp. 433–455, 2023. https://doi.org/10.1111/coin.12573.

[6] Lee, Y., J. Kim, and P. Kang, "LAnoBERT: System log anomaly detection based on BERT masked language model," Applied Soft Computing, vol. 146, Art. no. 110689, 2023. https://doi.org/10.1016/j.asoc.2023.110689.

[7] Liao, L., K. Zhu, J. Luo, and J. Cai, "LogBASA: Log anomaly detection based on system behavior analysis and global semantic awareness," International Journal of Intelligent Systems, vol. 2023, Art. no. 3777826, 2023. https://doi.org/10.1155/2023/3777826.

[8] Liu, Y., S. Ren, X. Wang, and M. Zhou, "Temporal logical attention network for log-based anomaly detection in distributed systems," Sensors, vol. 24, no. 24, Art. no. 7949, 2024. https://doi.org/10.3390/s24247949.

[9] Tian, G., N. Luktarhan, H. Wu, and Z. Shi, "CLDTLog: System log anomaly detection method based on contrastive learning and dual objective tasks," Sensors, vol. 23, no. 11, Art. no. 5042, 2023. https://doi.org/10.3390/s23115042.

[10] Wei, X., J. Wang, C. Sun, D. Towey, S. Zhang, W. Zuo, Y. Yu, R. Ruan, and G. Song, "Log-based anomaly detection for distributed systems: State of the art, industry experience, and open issues," Journal of Software: Evolution and Process, vol. 36, no. 8, Art. no. e2650, 2024. https://doi.org/10.1002/smr.2650

[11] Habiba, U. (2025, October 7). AI-driven assessment in TESOL: Adaptive feedback for personalized learning. TechRxiv. https://doi.org/10.36227/techrxiv.175987165.56867521.v1

[12] Islam, R. (2026, October 9). Financial statement analysis and profitability forecasting using machine learning models. SSRN. https://doi.org/10.2139/ssrn.7587418

[13] Kashem, M. A., Islam, M. R., Das, N., & Rokibuzzaman, M. (2026). An explainable decision intelligence architecture for secure data sharing and risk-aware resource governance. International Journal of Innovative Science and Research Technology, 11(8). https://doi.org/10.38124/ijisrt/26aug1015

[14] Habiba, U. (2025, October 7). Cross-cultural communication competence through technology-mediated TESOL. TechRxiv. https://doi.org/10.36227/techrxiv.175985896.67358551.v1

[15] Chowdhury, S. B. (2026). Dynamic business impact analysis and dependency-aware recovery optimization for mission-critical banking operations (Version 1) [Preprint]. Research Square. https://doi.org/10.21203/rs.3.rs-11541297/v1

[16] Rahman, M. H. (2026, October 24). Predictive analytics for hotel demand forecasting and dynamic revenue management. SSRN. https://doi.org/10.2139/ssrn.7663898

[17] Rahman, M. A. (2026). Cloud-based management information systems for integrated business operations and data-driven decision support. Preprints.org. https://doi.org/10.20944/preprints202610.2610.v1

[18] Rahman, M. A. (2026, October 23). Network performance monitoring and fault detection framework for enterprise IT infrastructure. SSRN. https://doi.org/10.2139/ssrn.7658978

[19] Arif, M. J. I. (2026). Digital twin-enabled dynamic production scheduling for concurrent multi-program manufacturing operations [Preprint]. Research Square. https://doi.org/10.21203/rs.3.rs-11378954/v1

[20] Bhuiyan, M. I. H. (2026). Predictive analytics framework for customer retention and service efficiency optimization in U.S. retail and digital banking systems [Preprint]. Preprints.org. https://doi.org/10.20944/preprints202610.1618.v1

[21] Arif, M. J. I. (2026). Lean-DMAIC integration for cycle time reduction in high-mix low-volume manufacturing environments: A framework for smart factory implementation. SSRN. https://doi.org/10.2139/ssrn.7613998

[22] Umar, M. (2026). Structural design and analysis framework for steel industrial structures under wind and seismic loading conditions. SSRN. https://doi.org/10.2139/ssrn.7608818

[23] Akter, L. (2026). Cognitive digital twin-driven explainable AI framework for adaptive detection of zero-day cyber threats [Preprint]. Research Square. https://doi.org/10.21203/rs.3.rs-11307626/v1

[24] Huq, S. N., Munna, A. A., Adiluzzaman, & Arif, A. (2026). Explainable process intelligence for workforce allocation and operational governance in digital healthcare systems. Saudi Journal of Engineering and Technology, 11(10), 920–929. https://doi.org/10.36348/sjet.2026.v11i10.003

[25] Hossain, M. R., Mumu, T. S., Ahmed, S., & Khalil, S. (2026). Explainable AI-driven decision intelligence for cyber-resilient enterprise operations under uncertainty. Saudi Journal of Engineering and Technology, 11(10), 911–919. https://doi.org/10.36348/sjet.2026.v11i10.002

[26] Adiluzzaman, A., Arif, A., Huq, S. N., & Munna, A. A. (2026). Human-centered explainable AI for adaptive resource allocation and decision governance in service organizations. Global Journal of Engineering and Technology Advances, 29(1), 168–180. https://doi.org/10.30574/gjeta.2026.29.1.0283

[27] Munna, A. A., Adiluzzaman, Arif, A., & Huq, S. N. (2026). Privacy-aware requirements traceability and risk intelligence for interoperable digital health information systems. International Journal of Science and Research Archive, 21(1), 276–287. https://doi.org/10.30574/ijsra.2026.21.1.1860