Hindawi
Security and Communication Networks
Volume 2023, Article ID 9761020, 1 page
https://doi.org/10.1155/2023/9761020

Retraction

Retracted: Classification and Prediction of Software Incidents Using Machine Learning Techniques

Security and Communication Networks

Received 5 December 2023; Accepted 5 December 2023; Published 6 December 2023

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] S. Ali, M. Adeel, S. Johar, M. Zeeshan, S. Baseer, and A. Irshad, "Classification and Prediction of Software Incidents Using Machine Learning Techniques," *Security and Communication Networks*, vol. 2021, Article ID 9609823, 16 pages, 2021.

Hindawi
Security and Communication Networks
Volume 2021, Article ID 9609823, 16 pages
https://doi.org/10.1155/2021/9609823

Research Article

Classification and Prediction of Software Incidents Using Machine Learning Techniques

Sikandar Ali,[1] **Muhammad Adeel**,[2] **Sumaira Johar**,[3] **Muhammad Zeeshan**,[4] **Samad Baseer**,[5] **and Azeem Irshad**[6]

[1]*Department of Computer Science and Technology, The University of Haripur, Haripur 22621, Khyber Pakhtunkhwa, Pakistan*
[2]*Department of Computer Science, School of Science, National Textile University, Faisalabad 37610, Punjab, Pakistan*
[3]*Department of Computer Science, University of Peshawar, Peshawar, Pakistan*
[4]*Institute of Computing, Kohat University of Science & Technology, Kohat, Pakistan*
[5]*Department of Computer System Engineering, University of Engineering and Technology, Peshawar 25000, Pakistan*
[6]*Department of Computer Science and Software Engineering, International Islamic University, Islamabad, Pakistan*

Correspondence should be addressed to Sikandar Ali; sikandar@cup.edu.cn, Muhammad Adeel; adeel@ntu.edu.pk, and Azeem Irshad; irshadazeem2@gmail.com

Received 13 October 2021; Revised 28 November 2021; Accepted 4 December 2021; Published 30 December 2021

Academic Editor: Muhammad Arif

An incident, in the perception of information technology, is an event that is not part of a normal process and disrupts operational procedure. This research work particularly focuses on software failure incidents. In any operational environment, software failure can put the quality and performance of services at risk. Many efforts are made to overcome this incident of software failure and to restore normal service as soon as possible. The main contribution of this study is software failure incidents classification and prediction using machine learning. In this study, an active learning approach is used to selectively label those data which is considered to be more informative to build models. Firstly, the sample with the highest randomness (entropy) is selected for labeling. Secondly, to classify the labeled observation into either failure or no failure classes, a binary classifier is used that predicts the target class label as failure or not. For classification, Support Vector Machine is used as a main classifier to classify the data. We derived our prediction models from the failure log files collected from the ECLIPSE software repository.

1. Introduction

In any particular system, failure befalls when the provided service no longer obeys the specified specifications [1]. Specifications are the agreed description of the system's functional behavior to provide expected service [1]. This definition applies to both software and hardware failures. According to Dalal and Chhillar [2], the most common software failure incidents on the web are pages not downloading properly due to sluggish response from the application server or application lack of compatibility with the browser, or it may be other performance issues such as slow load time, run time, or access time. Failures are of different types; i.e., not all the failures are fatal and some of them are even harmless and do not affect the functionality of the system. However, other failures are so fatal that they crash the whole system and make the system unavailable for specified services. But the types and levels of severity vary from software to software [2]. Faults, errors, and bugs in the software artifact are the ultimate cause of the software failure, which are the inappropriate process or step in the software artifact. Failures are the incapability of the software to perform the required action or in other words the deviation from required performance [2].

A failure or even a fractional failure of one service can cause other services that depend on it to break down. This incident can create a chain of service failures that propagates until it reaches critical components and causes the software to fail. According to Gray [3] in 1986, environment issues (e.g., cooling and power) and hardware issues (e.g., memory, network, and

disk) caused 32% of the incidents, which in 1999 decreased to 20% [4]. On the other hand software incidents increased from 26% to 40%. Some authors like Gray [4] even stated that 58% of the total incidents are software related. Incidents can be of different types, i.e., software incidents, hardware incidents, and technical incidents. This study focuses particularly on software incidents, which refers to the questionable behavior of the software. Software sometimes does not perform as it is expected due to many causes such as errors, bugs, and defects in it. These errors, bugs, and defects most of the time lead to software failure. In this study, we have extensively explored the software failure incidents, their causes, impacts, and the techniques proposed for their prediction. Keeping in view all these facts, this study builds a model for the prediction of software failure incidents. IT service providers are constantly seeking more efficient methods and implementations to increase the effectiveness and superiority of the process. IT Infrastructure Library (ITIL) is the widely used framework for IT services due to its best management guidelines. It provides the best guidelines on how to manage, develop, and maintain IT infrastructure. Above all, it also gives guidelines on improving the quality of the IT infrastructure. Organizations are investing heavily in operational environmental management applications. Software incidents in the operational environment are defined as unscheduled interruptions, which affect employees' productivity and also have impacts on the cost. To decrease the unscheduled interruptions and increase the performance, many incident management techniques are introduced. Software failure incidents on the web proposed that most of the failures occur during the system upgradation or the system maintenance and may sometimes be due to the system integration. There are many causes discussed in the relevant literature of software failures; such failures in software during operation are unavoidable. This causes the unavailability of the system which results in cost and dissatisfied customers and clients. These failures need to be reduced and removed for cost-effectiveness and the satisfaction of the customers. Most of the shared and agreed causes are inadequate testing or poor testing, flaws in documentation or the poor understanding of the system complexity, resource exhaustion, complex fault recovery routines, and system overload.

1.1. Contribution of the Study. The main contribution of this study is software failure incidents classification and prediction using machine learning. In this study, an active learning approach is used to selectively label those data that are considered to be more informative to build models. Firstly, the sample with the highest randomness (entropy) is selected for labeling. Secondly, to classify the labeled observation into either failure or no failure classes, a binary classifier is used that predicts the target class label as failure or not. For classification, Support Vector Machine is used as the main classifier of the data. We derived our prediction models from the failure log files collected from the ECLIPSE software repository.

1.2. Organization of the Paper. The remaining of the paper is organized into the following sections. Section 2 is based on related literature. Section 3 presents the classification and prediction method. Section 4 is results and analysis. Section 5 consists of results descriptions. Section 6 makes a discussion on the obtained results while Section 7 concludes the results and gives future directions.

2. Related Work

Efforts to foresee failures have been notable in recent decades. Failures, or prediction of failures, is a broad notion in software engineering that is not restricted to software failure. In both hardware and software, failure prediction techniques are widely used. These techniques are widely explored in the literature in hardware (e.g., satellite [5], distributed mission-critical systems [6], cluster computing systems [6], and telecommunication systems [7]). However, as software systems have become more complicated and there has been a greater requirement for reliability, the problems have migrated to the software [8]. Taherdoost et al. [9] surveyed to investigate the reasons for the failure and success of various information technology projects. They performed the survey, which included both technical and nontechnical aspects that are directly or indirectly related to the causes of failures, such as people and procedures.

Liang et al. [10] proposed an approach for predicting the failures in IBM's Blue Gene/L from the event logs generated by the systems. Event logs containing the records of the events generated by the system at different points of time are used for prediction. Sequential density is used to cover all the events at a single location. A lot of papers have been proposed in recent years analyzing high-performance computing (HPC) for prediction purposes. But many of these predictors are unable to use the required data for a long time; instead they use it only for short time. Furthermore, they required the new training phase after some time. This is the limitation of these predicting techniques. But many of the researchers tried to overcome these limitations such as Gu et al. [11]. In [11], they proposed two techniques; one is a meta-learning predictor to boost the accuracy and the second is the dynamic approach to collect and deal with the changing training set. The meta-learning predictor was proposed to provide a comparison between the rules-based and the statistical methods and further choose which of them is best for prediction purposes.

Nakka et al. [12] employ a hybrid technique to forecast failures in HPC systems, based on their usage as well as information from failure log files. This hybrid approach combines data mining classifications and signal analysis techniques. Another approach for failure prediction is proposed by Zheng and Yu [13] based on the reliability, availability, and serviceability (RAS) and job log files of the high computing system, i.e., Blue Gene/P. In comparison to other approaches, this approach does not predict the failures but filter those that do not affect the applications running on the system. A quite different approach for mining the interdependencies among the components of the HPC systems was proposed by Lou et al. [14]. They also used the log messages from the HPC system applications log messages to extract the information for mining the component's dependencies.

Gainaru et al. [15] suggested a new hybrid approach for predicting high-performance HPC failures using Blue Gene/

L log files combining signal analysis and data mining. They also discussed the problems and limitations attached to the failure prediction approaches. Xue et al. [16] talked about the failures in the cluster system and found the methods of collecting and processing data for failure prediction. They suggested a method for preprocessing the data in the log files. The researchers looked at rule-based classification, time series analysis, semi-Markov process models, and Bayesian network models as basic prediction methods. Gainaru et al. [17] presented a novel methodology for online failure prediction and showed that using this model, prediction is possible and easy for small systems. They showed the analysis of the feasibility of the online failure prediction methods on the Blue Waters system on pet scale machines.

Shalan and Zulkernine [18] proposed an approach for forecasting the failures in the software system during the system runtime. With the prediction of the failure, this approach also forecasts the occurrence of the modes in the software at the runtime. Pitakrat [19] also presented an online failure prediction approach called Hora. Hora is an online failure prediction approach based on the components of large-scale systems. This approach generates submodels for each component and then combines them using the interdependencies of the components. They used the Kieker framework and other tools such as WEKA and OPAD. Salfner et al. [20] discussed different online failure prediction approaches and developed a taxonomy that shows different approaches, their applications, and the results on implementation. Zhang et al. [9] proposed the new approach CASSANDRA for predicting runtime failures. The two current methodologies, design time and run time analysis techniques, were combined to create this new proposed approach. By developing an on-the-fly model of the future k-step global state space, they were able to forecast runtime problems.

Gupta et al. [21] surveyed the statistical method, time series analysis used for the prediction purposes. They studied the time series analysis and elaborated its working and the past work done using it for the software anomalies prediction. Liu et al. [22] proposed a hybrid version for short- and long-term software program failure time forecasting. This version consists of the SSA (singular spectrum analysis) and ARIMA for forecasting the time series of the software failure time. Fan et al. [23] used the time series modeling methods to analyze and forecast the failures in the construction equipment. They used time series approach to detect rules and patterns from huge amounts of data on equipment failures obtained through failure analysis and predictions for construction tools.

Among the many predictions strategies, time series analysis is common, but it carries some disadvantages too. A single message which is the source of the information in this approach is thought not to be enough for the failure prediction (Pinheiro et al., [24]). Li et al. [25] proposed the approach based on the time series analysis for detecting and estimating resource exhaustion time due to software aging. Time series ARMA model was developed to identify aging and predict resource exhaustion timeframes.

3. Methodology

We suggested a model for predicting software failure incidents using active learning and the Support Vector Machine (SVM) in this study. The dataset was subjected to active learning, which reduced the size of the dataset and picked a sample from it to serve as the training set for the SVM classifier. The sample was chosen because it had occurrences that were both unique and relevant in terms of training the classifier. The clustering technique is used to do active learning in this study. In our approach of clustering, we use k-mean clustering to feed the active learning process. The data was initially clustered using a k-mean clustering approach, and then the cluster representatives were utilized to label the data. These occurrences at the cluster's center were gathered and labeled by hand. These labeled data were utilized as the SVM classifier's training set, and classification was performed on it. After clustering, the training set appeared to be devoid of any repeated data and instances that had no useful information. Clustering was kept constant in this research, and no label propagation was used.

Clusters were analyzed in different ways to get the well-organized and the most "informative" sample from the dataset. The entropy of every cluster is measured and the clusters with higher entropy were considered to be the most informative. Clusters with diverse classes were also taken into consideration for having the best informative instances. To get the most informative set, different techniques were performed on the clusters. The final sample of the instances obtained was then labeled manually. The labeled training set was then used as the input to the SVM classifier. Sequential minimal optimization (SMO) algorithm of the SVM was selected to perform the classification. Data were split through "percentage splitter" and target class "level" from the attributes set was selected and started the classification procedure. This generated results which are shared.

4. Results and Analysis

4.1. WEKA 3.8.0

4.1.1. Objectives. Several conventional machine learning algorithms have been included in the program "Workbench" truncated WEKA by the Waikato team (Waikato Environment for Knowledge Analysis). With WEKA, the researcher can better utilize the Ml and extract knowledge from it that would otherwise be impossible to obtain from a vast quantity of data.

4.1.2. Documented Features. The WEKA contains a library of algorithms for perdition and data mining challenges. The software is written in Java 2 and contains a standardized interface to machine learning algorithms. WEKA makes use of the following data mining techniques.

(1) Selection of Attribute.

(2) Clustering.

(3) Classifiers (nonnumeric and both numeric).

(4) Rules for Association.

(5) Filters.

(6) Estimators.

4.2. Preprocessing of the Data. In this research log files of the eclipse, software is used as the dataset for the training and testing purposes of the predicting classifier. Log files generated during the last 3 months are collected from the repository of the software. As we know, WEKA uses mostly the ARFF format files and the CSV files; therefore, we transferred the data of the log files into the CSV file format. The dataset consists of 4 attributes, "Date and Time," "Source," "Event ID," and "Task Category".

(1) Date and Time attribute contains the time of the event occurrence.

(2) Source attribute mentions the node on which the event has been created such as the "software protection service failed," "Microsoft-Windows-DNS-Client," "TIMEOUT," "need updating," "Rtop service failed," "application error," and "ending window installer transaction'.

(3) Event ID contains the IDs for each type of event, but the same sources hold the same IDs even with different levels of severity.

(4) Task Category contains the category each task belongs to, such as "Event System," "none"," "−7," and "−212".

4.3. K-Mean Clustering of the Data.

(1) There are 100 instances and four attributes in our dataset.

(2) After loading the file in the WEKA, the data were subjected to clustering.

(3) Data were then clustered using the simple K-means clustering as shown in Figure 1.

(4) Three clusters of the hundred instances were created.

(5) Cluster 0, Cluster 1, and Cluster 2 are as shown in Figure 2.

4.4. Data Cluster Visualization. The data are clustered and visualized as shown in Figures 3 and 4 and in Table 1.

4.5. Entropy Calculation of the Clusters.

(1) In step one, we created the clusters of the data using the k-mean clustering technique.

(2) Three clusters were created for the "100" instances.

(3) The entropy of each cluster is then measured using the "entropy triangle" package installed in the WEKA.

(4) Cluster "2" is found to have the highest entropy as shown in Figure 5.

4.6. Cluster with Higher Entropy. Cluster 2 with 38 instances (Table 2) was found to have higher entropy due to the diversity of the data it contains. As discussed earlier, the higher the entropy is, the more the randomness of data is. The data in the cluster are assumed to be the most diverse and best for training and testing the classifier due to its uncertainty for the labels.

4.7. Evaluation of the Entropy Calculation (Manual Labeling). (1) The entropy of Cluster 2 is the highest among all the clusters.

(2) The highest entropy means it contains diverse data.

(3) Cluster 1 has more instances than Cluster 2, but it does not have a variety of classes.

(4) Table 1 shows that Cluster 1 has the "warning" class instances more than any other class.

(5) Figure 5 shows that Cluster 2 with higher entropy has a variety of classes.

(6) A comparison of both is shown in Tables 3 and 4.

(7) Both were manually labeled to evaluate the entropy calculation.

(8) A new attribute with the name "LEVEL" is added to the dataset. The level attribute contains the data of the level of the severity of the generated log file against any event that occurred in the software. The levels can be of 4 types in our dataset "WARNING," "ERROR," "FAILED," and "STATUS-OK".

5. Results Descriptions

We have started our process of model building with the "Explorer" window. Explorer window toolbar items "preprocessing," "Classify," and "Cluster" are used to build the model. Figure 1 depicts our dataset file with the load in WEKA, WEKA calculated the attributes, instances, weighted averages, uniqueness, and the classes. The right corner of the window gave the graphical visualization of our instances. Figure 2 is the type of table containing the summary of the clustering. Clustering of the whole dataset is performed, and 3 clusters were created.

Tables 5 and 6 are the summaries of the k means clustering performed in the WEKA. Table 7 shows the model and evaluation training on set. Figures 3 and 4 are the graphical visualizations of the k-mean clustering. The clusters of the 100 instances saved for labeling are shown in Table 1. Table 2 is the Cluster 2 data chosen for labeling. Figure 5 shows that Cluster 2 with higher entropy has a variety of classes. Table 3 is Cluster 1, which shows the repetition of the same class. Table 4 is Cluster 2 for labeling. Table 8 is the Cluster 2 data chosen for labeling. Figure 6 is the extracted set of instances from the whole dataset chosen for labeling and is the most informative dataset for training the classifier. The dataset was labeled manually and then was expected to be the most useful subset for training the classifier. Table 9 manual labeling of Cluster 2. Three clusters can be seen in the window with different numbers of the data points in Table 10. Table 11 shows the detailed accuracy by

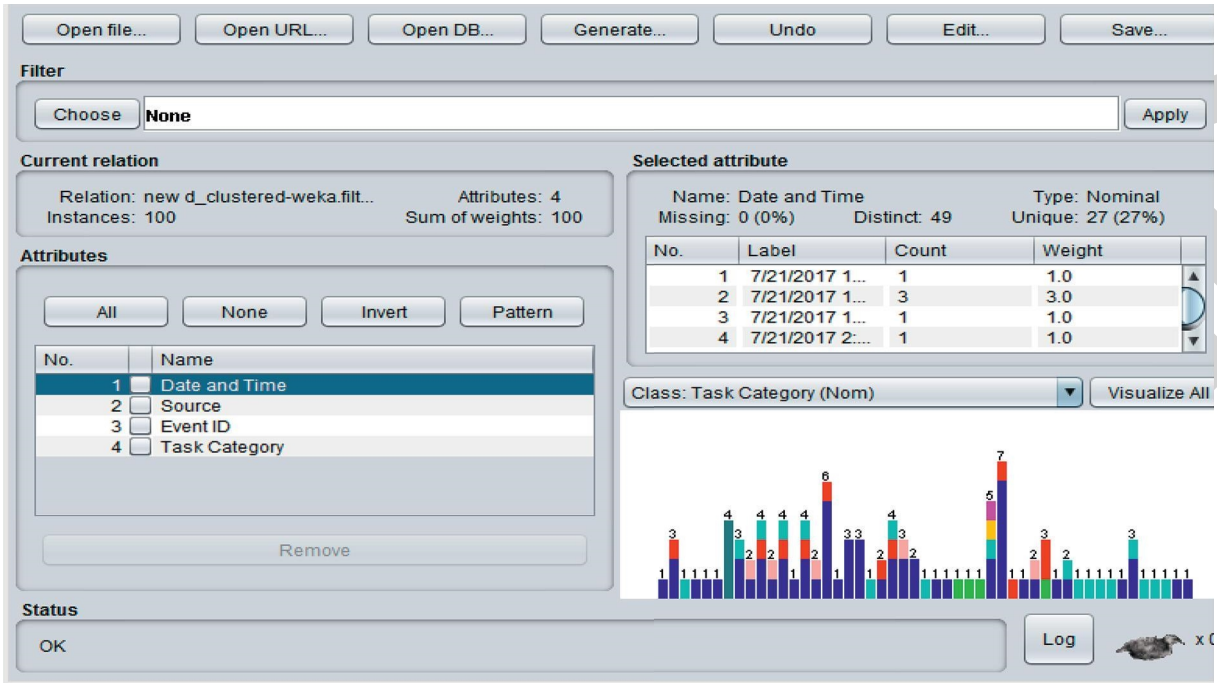

Figure 1: Clustered data.

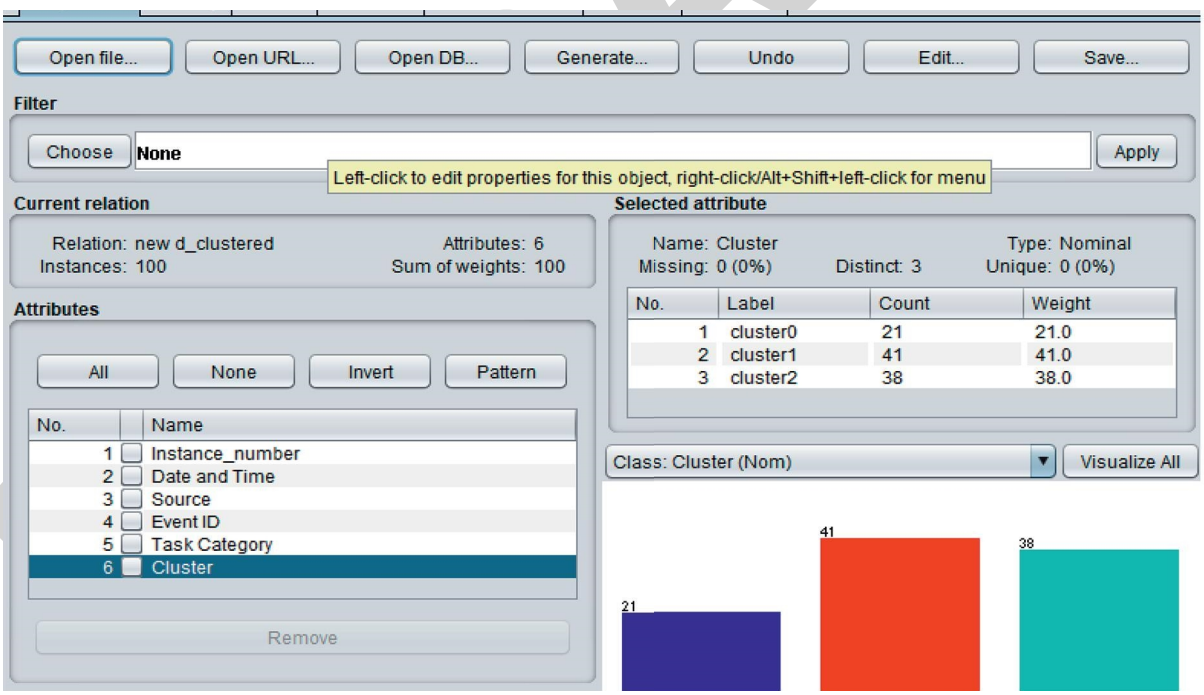

Figure 2: Clustered data.

class and Table 12 is the confusion matrix. Figure 5 is the entropy calculation of the clusters. The entropy of each cluster is calculated and the clusters with high entropy and the large size are chosen for labeling and considered the most informative cluster for training the classifier.

Figures 7 and 8 are the summary of the classification performed on the selected cluster. Cluster 2 is chosen because it has high entropy and the number of instances it has is higher than the other clusters. With percentage, the False Positive (FP) Rate, True Positive (TP) Rate, Precision Recall,

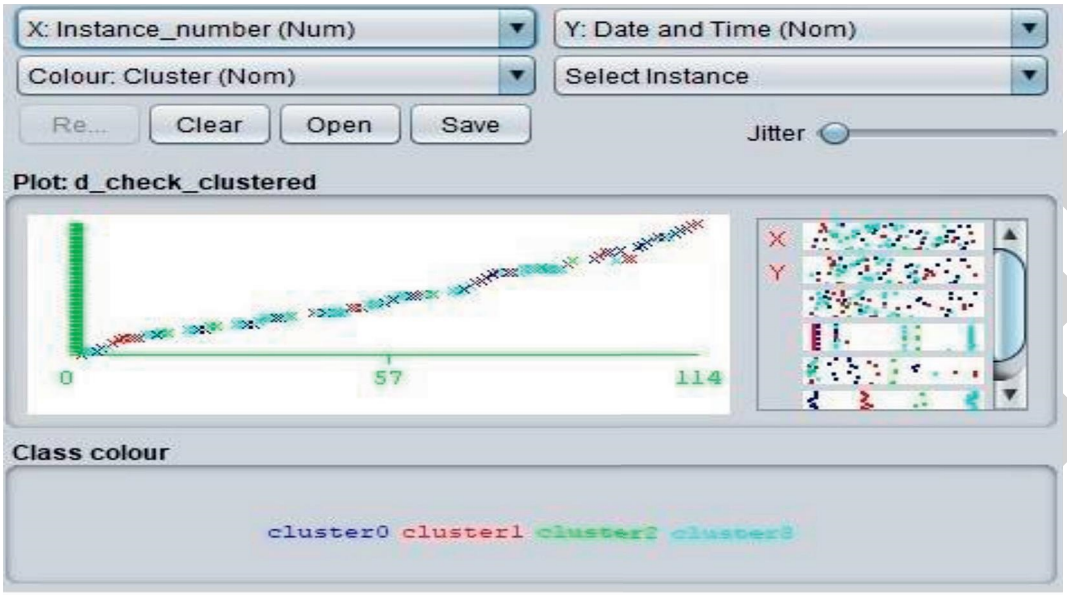

Figure 3: Clustered and visualized data.

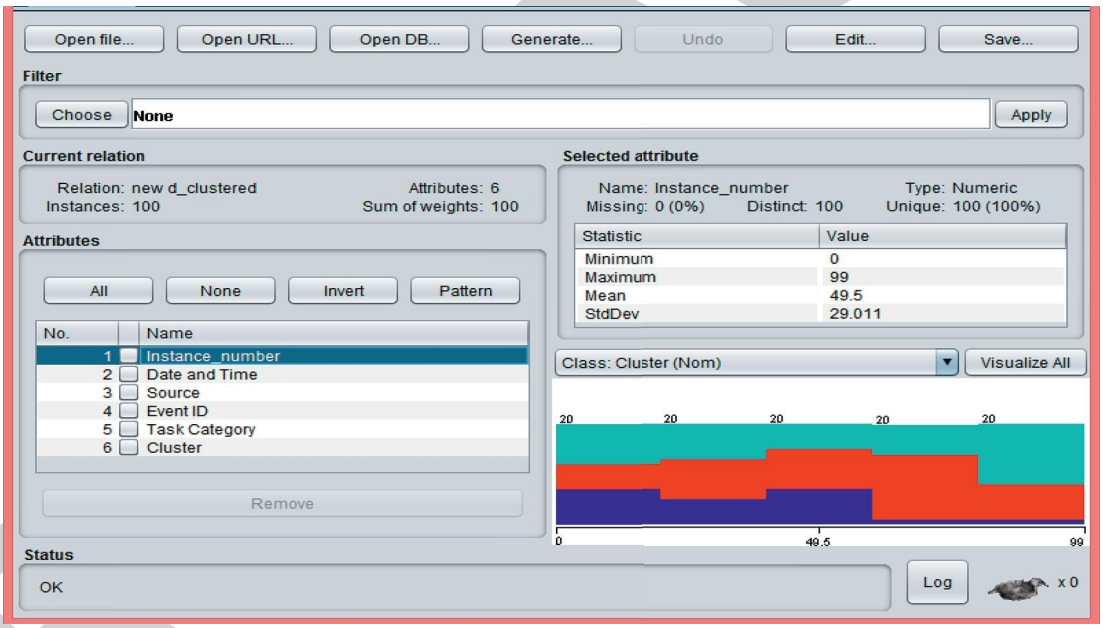

Figure 4: Clustered and visualized data.

F-measure, ROC, MCC, and PRC Area for each characteristic are displayed in the test summary as well as the correctly classified and nonclassified occurrences.

6. Discussion

Our model had an accuracy of 84 percent, properly categorizing the vast majority of the occurrences with only two exceptions. The test model, as shown in Figure 8, displayed the model's detailed accuracy measure using terminology such as *F*-measure and Precision Recall. Some terminology, such as FP Rate, TP Rate, *F*-measure, Precision Recall, MCC, ROC, and PRC Area must be understood before addressing the measure.

6.1. True Positive (TP). Positive values are both observed and forecasted to be positive. In our model, the TPR for the "failed" level is 1.000; for status ok it is 0.5; for error it is 1.00; and for warning it is 0.833.

6.2. False Positive (FP). When a negative value is observed, a positive forecast result is obtained. In our situation, FP stands for failure level which is 0.08 and for error level it is 0.111.

Table 1: Clustered and visualized data.

Number	Instance_	Source	Event ID	Task Category	Clusters
1	0	ReasonBy	10	None	Cluster1
2	1	App-configuration	7000	None	Cluster0
3	2	App-configuration	7009	None	Cluster0
4	3	Rtop service	1002	Address	Cluster2
5	4	Rtop service	219	−212	Cluster2
6	5	Rtop service	4001	None	Cluster1
7	6	ReasonBy	10	None	Cluster1
8	7	Network	4	None	Cluster1
9	8	Ending	37	−7	Cluster2
10	9	Ending	37	−7	Cluster2
11	10	Rtop service	37	−7	Cluster2
12	11	Rtop service	37	−7	Cluster2
13	12	App-configuration	7000	None	Cluster0
14	13	App-configuration	7009	None	Cluster0
15	14	Ending	219	−212	Cluster2
16	15	Rtop service	4001	None	Cluster1
17	16	Ending	4621	Event Sys1	Cluster0
18	17	App-configuration	7000	None	Cluster0
19	18	App-configuration	None	Cluster0	
20	19	Rtop service	1002	Address	Cluster2
21	20	Rtop service	219	−212	Cluster2
22	21	Ending	4001	None	Cluster1
23	22	Rtop service	4001	None	Cluster1
24	23	ReasonBy	10	None	Cluster1
25	24	App-configuration		None	Cluster0
26	25	Rtop service	1002	Address	Cluster2
27	26	Ending	219	−212	Cluster2
28	27	ReasonBy	10	None	Cluster1
29	26	App-configuration	7000	None	Cluster0
30	31	App-configuration	7009	None	Cluster0
31	30	Rtop service	1002	Address	Cluster2
32	31	Software	219	−212	Cluster2
33	32	Rtop service	4001	None	Cluster1
34	33	Rtop service	4621	Event Sys1	Cluster2
35	34	Rtop service	4001	None	Cluster1
36	35	Network	4	None	Cluster1
37	36	Rtop service	1002	Address	Cluster2
38	37	App-configuration	7000	None	Cluster0
39	38	App-configuration	7009	None	Cluster0
40	39	Rtop service	4001	None	Cluster1
41	40	ReasonBy	10	None	Cluster1
42	41	Network	4	None	Cluster1
43	42	Software	219	None	Cluster1
44	43	Software	219	None	Cluster2
45	44	App-configuration	7000	None	Cluster0
46	45	Rtop service	4001	None	Cluster1
47	46	Ending	4621	Event Sys1	Cluster0
48	47	Software	219	-212	Cluster2
16	15	Rtop service	4001	None	Cluster1
50	49	Need upd	1002	Address	Cluster2
51	50	Ending	4621	Event Sys1	Cluster0
52	51	App-configuration	7000	None	Cluster0
53	52	Microsoft	1002	Address	Cluster2
54	53	Initialization	219	−212	Cluster2
55	54	App-configuration	7000	None	Cluster0
56	55	App-configuration	7009	None	Cluster0
57	56	Rtop service	4001	None	Cluster1
58	57	ReasonBy	10	None	Cluster1
59	58	Network	4	None	Cluster1
60	59	Software	219	−212	Cluster2

Table 1: Continued.

Number	Instance_	Source	Event ID	Task Category	Clusters
61	60	Rtop service	4001	None	Cluster1
62	61	ReasonBy	10	None	Cluster1
63	62	Application	1000	−100	Cluster2
64	63	Application	1000	−100	Cluster2
65	64	Rtop service	4001	None	Cluster1
66	65	Need upd	4	None	Cluster2
67	66	Software	1101	Event Sys1	Cluster2
68	67	ReasonBy	10	None	Cluster1
69	68	Application	1000	−100	Cluster2
70	69	Application	1000	−100	Cluster2
71	70	Ending	1014	None	Cluster2
72	71	Software	1014	None	Cluster0
73	72	Software	1002	Address	Cluster2
74	73	ReasonBy	10	None	Cluster1
75	74	Ending	1014	None	Cluster2
76	75	Microsoft	1014	None	Cluster1
77	76	Microsoft	1014	None	Cluster1
78	77	Need upd	4	None	Cluster1
79	78	Network	4	None	Cluster1
80	79	Software	4621	Event Sys1	Cluster0
81	80	Rtop service	4001	None	Cluster1
82	81	Microsoft	52	None	Cluster1
83	82	Software	4	None	Cluster0
84	83	Software	5	None	Cluster0
85	84	Duplicate	4001	None	Cluster1
86	85	Software	219	−212	Cluster2
87	86	Rtop service	4001	None	Cluster1
88	87	Rtop service	4621	Event Sys1	Cluster2
89	88	Rtop service	4001	None	Cluster1
90	89	Network	4	None	Cluster1
91	90	Rtop service	1002	Address	Cluster2
92	91	App-configuration	7000	None	Cluster0
93	92	Network	4	None	Cluster1
94	93	Rtop service	4001	None	Cluster1
95	94	Microsoft	4001	None	Cluster0
96	95	ReasonBy	10	None	Cluster1
97	96	Ending	1014	None	Cluster2
98	97	Microsoft	1014	None	Cluster1
99	98	Network	4	None	Cluster1
100	99	Need upd	4	None	Cluster1
101	100	Rtop service	37	−7	Cluster0
102	101	Ending	219	−212	Cluster2
103	102	Network	4	None	Cluster1
104	103	Software	4621	Event Sys1	Cluster0

6.3. Precision. Precision is calculated as the number of accurately identified positive events divided by the total number of occurrences predicted. As indicated in Figure 9, the precision rate for failure is 0.5, status ok is 1.000, error is 0.8, and warning is 1.00.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}. \quad (1)$$

6.4. Recall. The number of accurately projected positive values divided by the total number of observations is called Recall. As previously stated, recall is the proportion of true positive observations to total observations. In our situation, the Recall is "1.000" for failure, "0.5" for status ok, "1.000" for error, and "0.833" for warning.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False Negative}}. \quad (2)$$

6.5. F-Measure. As indicated in Figure 10, the *F*-measure for failure is "0.667," status ok is "0.677," error is "0.84," and warning is "0.85" in our model.

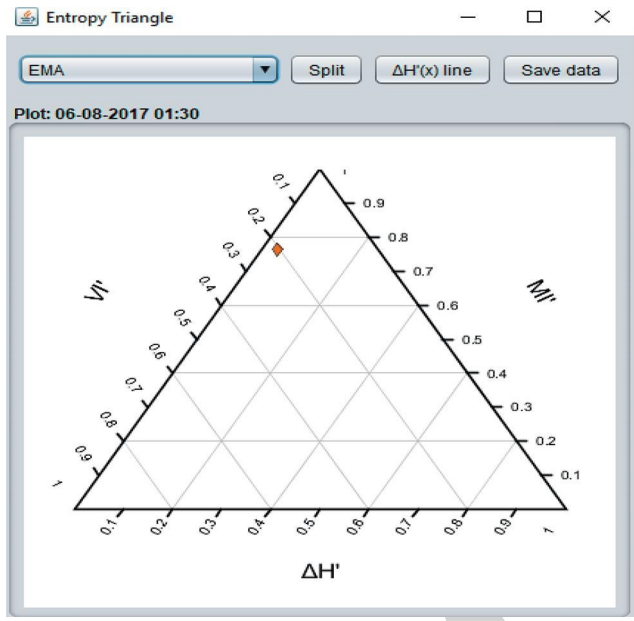

Figure 5: Entropy triangle.

Table 2: Clusters with higher entropy.

Number	Instance_	Source	Event ID	Task category	Clusters
4	3	Rtop service	1002	Address	Cluster2
5	4	Rtop service	219	−212	Cluster2
9	8	Ending	37	−7	Cluster2
10	9	Ending	37	−7	Cluster2
11	10	Rtop service	37	−7	Cluster2
12	11	Rtop service	37	−7	Cluster2
15	14	Ending	219	−212	Cluster2
20	19	Rtop service	1002	Address	Cluster2
21	20	Rtop service	219	−212	Cluster2
26	25	Rtop service	1002	Address	Cluster2
27	26	Ending	219	−212	Cluster2
31	30	Rtop service	1002	Address	Cluster2
32	31	Software	219	−212	Cluster2
34	33	Rtop service	4621	Event Sys1	Cluster2
37	36	Rtop service	1002	Address	Cluster2
48	47	Software	219	−212	Cluster2
50	49	Need upd	1002	Address	Cluster2
53	52	Microsoft	1002	Address	Cluster2
54	53	Initialization	219	−212	Cluster2
60	59	Software	219	−212	Cluster2
63	62	Application	1000	−100	Cluster2
64	63	Application	1000	−100	Cluster2

Table 3: Cluster 1 labels.

Number.	Instance	Source	Event ID	Task category	Clusters
4	3	Rtop service	1002	Address	Failed
5	4	Rtop service	219	−212	Failed
9	8	Ending	37	−7	Status ok
10	9	Ending	37	−7	Status ok
11	10	Rtop service	37	−7	Failed
12	11	Rtop service	37	−7	Failed
15	14	Ending	219	−212	Status ok
20	19	Rtop service	1002	Address	Error
21	20	Rtop service	219	−212	Warning
26	25	Rtop service	1002	Address	Warning
27	26	Ending	219	−212	Status ok
31	30	Rtop service	1002	Address	Error
32	31	Software	219	−212	Warning
34	33	Rtop service	4621	Event Sys1	Error
37	36	Rtop service	1002	Address	Error
48	47	Software	219	−212	Failed
50	49	Need upd	1002	Address	Error
53	52	Microsoft	1002	Address	Error
54	53	Initialization	219	−212	Warning
60	59	Software	219	−212	Warning
63	62	Application	1000	−100	Error
64	63	Application	1000	−100	Error

Table 4: Cluster 2 labels.

Number	Instance_	Source	Event ID	Task category	Clusters
4	3	Rtop service	1002	Address	Failed
5	4	Rtop service	219	−212	Failed
9	8	Ending	37	−7	Status ok
10	9	Ending	37	−7	Status ok
11	10	Rtop service	37	−7	Failed
12	11	Rtop service	37	−7	Failed
15	14	Ending	219	−212	Status ok
20	19	Rtop service	1002	Address	Error
21	20	Rtop service	219	−212	Warning
26	25	Rtop service	1002	Address	Warning
27	26	Ending	219	−212	Status ok
31	30	Rtop service	1002	Address	Error
32	31	Software	219	−212	Warning
34	33	Rtop service	4621	Event Sys1	Error
37	36	Rtop service	1002	Address	Error
39	38	Microsoft	1014	None	Warning
41	40	Microsoft	1014	None	Warning
42	41	Need upd	4	None	Warning
45	44	Network	4	None	Warning
46	45	Software	4621	Event Sys1	Failed
47	46	Microsoft	52	None	Warning
48	47	Software	219	−212	Failed
50	49	Need upd	1002	Address	Error
53	52	Microsoft	1002	Address	Error
54	53	Initialization	219	−212	Warning
60	59	Software	219	−212	Warning
63	62	Application	1000	−100	Error
64	63	Application	1000	−100	Error

Table 5: Summary of K-mean clustering.

k-Means
======
Number of iterations: 4
Within cluster sum of squared errors: 186.97500542403395

Initial starting points (random):			
Cluster 0: "7/20/2017 23: 10"	"App-configuration failed"	7009	None
Cluster 1: "7/18/2017 23: 09"	"Need updation"	6008	None
Cluster 2: "7/20/2017 23: 34"	"Ending window installer transaction"	37	−7

Missing values globally replaced with mean/mode
Final cluster centroids:

		Clusters #		
Attribute	Full data (100)	Cluster 0 (21.0)	Cluster 1 (41.0)	Cluster 2 (38.0)
Date and time	7/17/2017 12:07	7/21/2017 11:15	7/17/2017 12:07	7/20/2017 23:34
Source	Software protection service failed	App configuration failed	Software protection service failed	Rtop service failed
Event ID	2204.01	6617.4286	1477.0244	549.3947
Task category	None	None	None	−212

Time taken to build the model (full training data): 0.02 seconds

Table 6: Summary of k-mean clustering.

Attribute	Full data (100.0)	0 (21.0)	1 (41.0)	2 (38.0)
Instance number	49.5	34.6667	54.0732	52.7632
Date and time	7/17/2017 12:07	7/21/2017 11:55	7/17/2017 12:07	7/20/2017 23:34
Source	Software protection service failed	App-configuration failed	Software protection service failed	Rtop service failed
Event ID	2204.01	6617.4286	1477.0244	549.3947
Task category	None	None	None	-212
Cluster	Cluster1	Cluster0	Cluster1	Cluster2

Table 7: Model and evaluation on training set.

Clustered instances	
0	21 (21%)
1	41 (41%)
2	38 (38%)

Table 8: Selection of clusters.

Number.	Instance	Source	Event ID	Task category	Clusters
4	3	Rtop service	1002	Address	Cluster2
5	4	Rtop service	219	−212	Cluster2
9	8	Ending	37	−7	Cluster2
10	9	Ending	37	−7	Cluster2
11	10	Rtop service	37	−7	Cluster2
12	11	Rtop service	37	−7	Cluster2
15	14	Ending	219	−212	Cluster2
20	19	Rtop service	1002	Address	Cluster2
21	20	Rtop service	219	−212	Cluster2
26	25	Rtop service	1002	Address	Cluster2
27	26	Ending	219	−212	Cluster2
31	30	Rtop service	1002	Address	Cluster2
32	31	Software	219	−212	Cluster2
34	33	Rtop service	4621	Event Sys1	Cluster2
37	36	Rtop service	1002	Address	Cluster2

Table 8: Continued.

Number.	Instance	Source	Event ID	Task category	Clusters
48	47	Software	219	−212	Cluster2
50	49	Need upd	1002	Address	Cluster2
53	52	Microsoft	1002	Address	Cluster2
54	53	Initialization	219	−212	Cluster2
60	59	Software	219	−212	Cluster2
63	62	Application	1000	−100	Cluster2
64	63	Application	1000	−100	Cluster2

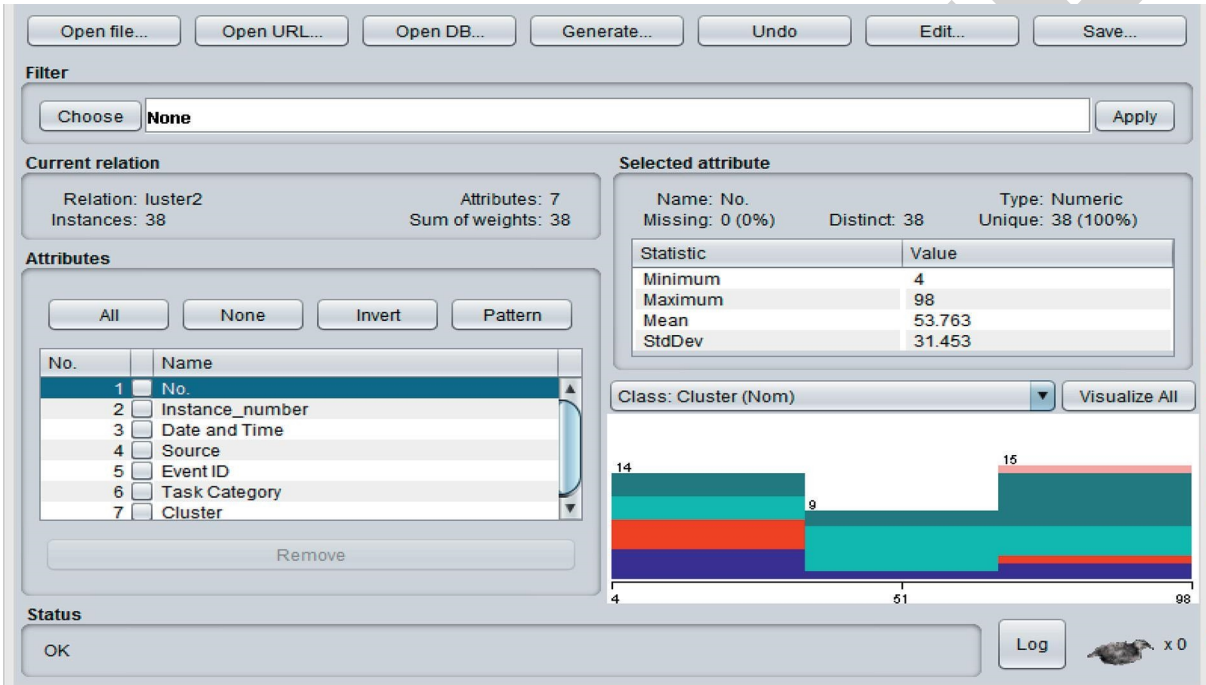

Figure 6: Selection of clusters.

Table 9: Selection of clusters.

Number.	Instance_	Source	Event ID	Task category	Clusters
4	3	Rtop service	1002	Address	Failed
5	4	Rtop service	219	−212	Failed
9	8	Ending	37	−7	Status ok
10	9	Ending	37	−7	Status ok
11	10	Rtop service	37	−7	Failed
12	11	Rtop service	37	−7	Failed
15	14	Ending	219	−212	Status ok
20	19	Rtop service	1002	Address	Error
21	20	Rtop service	219	−212	Warning
26	25	Rtop service	1002	Address	Warning
27	26	Ending	219	−212	Status ok
31	30	Rtop service	1002	Address	Error
32	31	Software	219	−212	Warning
34	33	Rtop service	4621	Event Sys1	Error
37	36	Rtop service	1002	Address	Error
39	38	Microsoft	1014	None	Warning
41	40	Microsoft	1014	None	Warning
42	41	Need upd	4	None	Warning
45	44	Network	4	None	Warning
46	45	Software	4621	Event Sys1	Failed
47	46	Microsoft	52	None	Warning

Table 9: Continued.

Number.	Instance_	Source	Event ID	Task category	Clusters
48	47	Software	219	−212	Failed
50	49	Need upd	1002	Address	Error
53	52	Microsoft	1002	Address	Error
54	53	Initialization	219	−212	Warning
60	59	Software	219	−212	Warning
63	62	Application	1000	−100	Error
64	63	Application	1000	−100	Error

Table 10: Clustered instances.

Clustered instances	
0	21 (21%)
1	41 (41%)
2	38 (38%)

Table 11: Detailed accuracy.

	TP rate	FP rate	Precision	Recall	*F*-measure	MCC	ROC area	PRC area	Class
	1.000	0.083	0.500	1.000	0.667	0.667	0.958	0.500	Failed
	0.500	0.000	1.000	0.500	0.667	0.667	0.977	0.833	Status ok
	1.000	0.111	0.800	1.000	0.889	0.843	0.944	0.800	Error
	0.833	0.000	1.000	0.833	0.909	0.854	0.917	0.910	Warning
	0.000	0.000	0.000	0.000	0.000	0.000	?	?	Critical
Weighted AVG	0.846	0.041	0.900	0.846	0.847	0.810	0.938	0.833	

Table 12: Confusion matrix.

a	b	c	d	E	<--	Classified as
1	0	0	0	0	$\|a =$	Failed
0	1	1	0	0	$\|b =$	Status ok
0	0	4	0	0	$\|c =$	Error
1	0	0	5	0	$\|d =$	Warning

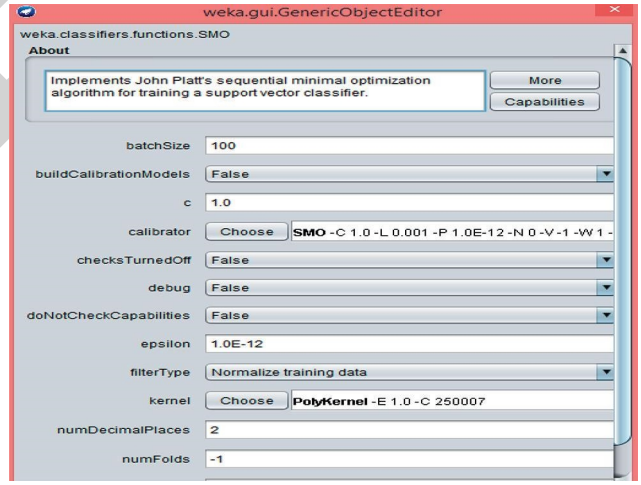

Figure 7: Kernel selection.

```
=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances          11               84.6154 %
Incorrectly Classified Instances         2               15.3846 %
Kappa statistic                          0.7739
Entropy Modulated Accuracy               0.7415
Normalized Information Transfer factor   0.4947
Mean absolute error                      0.2492
Root mean squared error                  0.3305
Relative absolute error                 82.9352 %
Root relative squared error             85.6331 %
Total Number of Instances               13

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.083    0.500      1.000    0.667      0.677    0.958     0.500     failed
                 0.500    0.000    1.000      0.500    0.667      0.677    0.977     0.833     status ok
                 1.000    0.111    0.800      1.000    0.889      0.843    0.944     0.800     error
                 0.833    0.000    1.000      0.833    0.909      0.854    0.917     0.910     warning
                 0.000    0.000    0.000      0.000    0.000      0.000    ?         ?         critial
Weighted Avg.    0.846    0.041    0.900      0.846    0.847      0.810    0.938     0.833

=== Confusion Matrix ===

 a b c d e   <-- classified as
 1 0 0 0 0 | a = failed
 0 1 1 0 0 | b = status ok
 0 0 4 0 0 | c = error
 1 0 0 5 0 | d = warning
 0 0 0 0 0 | e = critial
```

Figure 8: Model for test data.

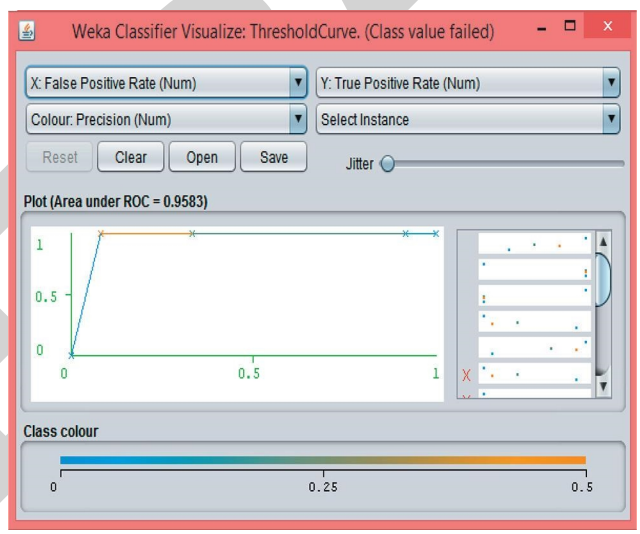

Figure 9: Failure precision (plot under ROC).

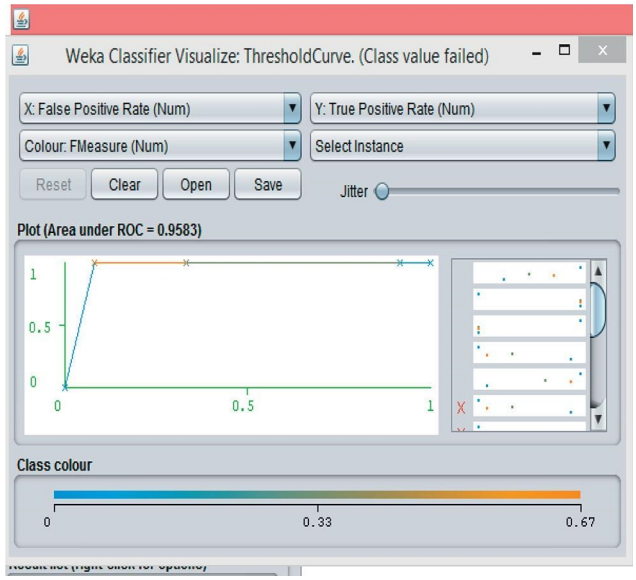

Figure 10: Failure *F*-measure (plot under ROC).

$$F\text{measure} = \frac{2(\text{Recall} \times \text{Precision})}{(\text{Recall} \times \text{Precision})}. \quad (3)$$

6.6. Confusion Matrix. The confusion matrix, also known as the error matrix (Table 12), is a visual representation of the technique's or algorithm's performance. The projected cases are in the rows, whereas the actual instances are in the columns.

7. Conclusion and Future Work

Every failure, in general, is critical in terms of both security and cost. Forecasting techniques can be used to recognize enhanced and better maintenance schedules. Failure forecasts aid in the prediction of maintenance times, reducing both costs and security richness. This research provided a model for forecasting failures based on machine learning methodologies and techniques, active learning via clustering, and SVM classification of selected examples. Although SVM calculations are known to be capable of predicting, it is unclear how to choose the parameter values that will provide a satisfactory result. However, modeling a function for transformative computations to be used in determining requirements for the combination of a large number of possible outcomes is difficult. The goal of this study is to predict software faults in order to optimize maintenance schedules and demonstrate and predict sophisticated software system failures. We used two machine learning algorithms to do this. We gathered log papers with the four qualities and 100 examples. We used a dynamic learning method to reduce the number of variables.

Grouping and SVM were used to display event-driven error log records. Review, exactness, *F*-measure, and precision were used to describe the models' quality. Our findings show that active learning and SVM are the most commonly used techniques. Expecting all failures to keep a strategic distance from our showing techniques may result in a request for a significant modification in framework accessibility. The goal is to achieve the best execution with the most useful information.

Data Availability

The data will be available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] J. C. Laprie, *Dependability: Basic Concepts and Terminology*, Springer, Vienna, Austria, pp. 3–245, 1992.

[2] S. Dalal and R. S. Chhillar, "Empirical study of root cause analysis of software failure," *ACM SIGSOFT-Software Engineering Notes*, vol. 38, no. 4, pp. 1–7, 2013.

[3] J. Gray, "Why do computers stop and what can be done about it?" in *Proceedings of the Symposium on Reliability in Distributed Software and Database Systems*, pp. 3–12, Los Angeles, CA, USA, January 1986.

[4] J. Gray, "A census of Tandem system availability between 1985 and 1990," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 409–418, 1990.

[5] S. Bottone, D. Lee, M. O'Sullivan, and M. Spivack, "Failure prediction and diagnosis for satellite monitoring systems using bayesian networks," in *Proceedings of the 2008 IEEE Military Communications Conference (MILCOM)*, pp. 1–7, IEEE, San Diego, CA, USA, November 2008.

[6] Y. Li and Z. Lan, "Exploit failure prediction for adaptive fault-tolerance in cluster computing," in *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, vol. 1, p. 8, IEEE, Singapore, May 2006.

[7] R. Baldoni, G. Lodi, L. Montanari, G. Mariotta, and M. Rizzuto, "Online black-box failure prediction for mission critical distributed systems," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*, pp. 185–197, Springer, Magdeburg, Germany, 2012, September.

[8] F. Salfner and S. Tschirpke, "Error log processing for accurate failure prediction," in *Proceedings of the First USENIX Conference on Analysis of System Logs (WASL'2008)*, San Diego, CA, USA, December 2008.

[9] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys*, vol. 42, no. 3, pp. 1–42, 2010.

[10] H. Taherdoost and Keshavarzsaleh, "A theoretical review on IT project success/failure factors and evaluating the associated risks," in *Proceedings of the 14th International Conference on Telecommunications and Informatics*, Sliema, Malta, August 2015.

[11] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM bluegene/l event logs," in *Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM)*, pp. 583–588, Bandung, Indonesia, October 2007.

[12] Z. Lan, J. Gu, Z. Zheng, R. Thakur, and S. Coghlan, "A study of dynamic meta-learning for failure prediction in large-scale systems," *Journal of Parallel and Distributed Computing*, vol. 70, no. 6, pp. 630–643, 2010.

[13] N. Nakka, A. Agrawal, and A. Choudhary, "Predicting node failure in high performance computing systems from failure and usage logs," in *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pp. 1557–1566, IEEE, Anchorage, AK, USA, May 2011.

[14] Z. Zheng, L. Yu, W. Tang et al., "Co-analysis of RAS log and job log on Blue Gene/P," in *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, pp. 840–851, Anchorage, AK, USA, May 2011.

[15] J.-G. Lou, Q. Fu, Y. Wang, and J. Li, "Mining dependency in distributed systems through unstructured logs analysis," *ACM SIGOPS - Operating Systems Review*, vol. 44, no. 1, pp. 91–96, 2010.

[16] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Failure prediction for HPC systems and applications," *International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 273–282, 2013.

[17] Z. Xue, X. Dong, S. Ma, and W. Dong, "A survey on failure prediction of large-scale server clusters," in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, pp. 733–738, Qingdao, China, August 2007.

[18] A. Gainaru, M. S. Bouguerra, F. Cappello, M. Snir, and W. Kramer, "Navigating the blue waters: online failure prediction in the petascale era," Technical Report, ANL/MCS-P5219–1014, Argonne National Laboratory, Illinois, IL, USA, 2013.

[19] A. Shalan and M. Zulkernine, "Runtime prediction of failure modes from system error logs," in *Proceedings of the 2013 18th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pp. 232–241, Singapore, July 2013.

[20] T. Pitakrat, "Hora: online failure prediction framework for component-based software systems based on kieker and palladio,"vol. 27-29, pp. 39–48, in *Proceedings of the Symposium on Software Performance: Joint Kieker/Palladio Days 2013*, vol. 27-29, pp. 39–48, Kieker/Palladio Days 2013, Karlsruhe, Germany, November 2013.

[21] P. Zhang, H. Muccini, A. Polini, and X. Li, "Run-time systems failure prediction via proactive monitoring," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pp. 484–487, Lawrence, KS, USA, November 2011.

[22] B. R. Ashima Gupta, "Prediction of software anomalies using time series analysis – a recent study," *International Journal of Advances in Computer Science and Cloud Computing*, vol. 2, no. 3, pp. 101–108, 2013.

[23] G. Liu, D. Zhang, and T. Zhang, "Software reliability forecasting: singular spectrum analysis and ARIMA hybrid model," in *Proceedings of the 2015 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pp. 111–118, Nanjing, China, September, 2015.

[24] Q. Fan and H. Fan, "Reliability analysis and failure prediction of construction equipment with time series models," *Journal of Advanced Management Science*, vol. 3, no. 3, pp. 202–210, 2015.

[25] E. Pinheiro, W. D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies*, pp. 17–23, San Jose, CA, USA, February 2007.