Independent University **Bangladesh (IUB)**

IUB Academic Repository

Research Articles 2023

2023

Develop a System to Analyze Logs of a Given System Using Machine Learning

Hasan, Md. Tarek

Independent University, Bangladesh (IUB)

https://ar.iub.edu.bd/handle/11348/575

Downloaded from IUB Academic Repository

Draft Version

Develop a System to Analyze Logs of a Given System Using Machine Learning

Md. Tarek Hasan, Farzana Sadia, Mahady Hasan and M. Rokonuzzaman

Abstract Software error detection is a critical aspect of software development. However, due to the lack of time, budget, and workforce, testing applications can be challenging, and in some cases, bug reports may not make it to the final stage. Additionally, a lack of product domain knowledge can lead to misinterpretation of calculations, resulting in errors. To address these challenges, early bug prediction is necessary to develop error-free and efficient applications. In this study, the author proposed a system that uses machine learning to analyze system error logs and detect errors in real time. The proposed system leverages imbalanced data sets from live servers running applications developed using PHP and Codeigniter. The system uses classification algorithms to identify errors and suggests steps to overcome them, thus improving the software's quality, reliability, and efficiency. Our approach addresses the challenges associated with large and complex software where it can be difficult to identify bugs in the early stages. By analyzing system logs, we demonstrate how machine learning classification algorithms can be used to detect errors and improve system performance. Our work contributes to a better understanding of how machine learning can be used in real-world applications and highlights the practical benefits of early bug prediction in software development.

Md. Tarek Hasan
Independent University, Bangladesh, Plot 16 Block B, Bashundhara R/A, Dhaka, Bangladesh. e-mail: 2031276@iub.edu.bd

Farzana Sadia
Independent University, Bangladesh, Plot 16 Block B, Bashundhara R/A, Dhaka, Bangladesh. e-mail: fsbornasets@iub.edu.bd

Mahady Hasan
Independent University, Bangladesh, Plot 16 Block B, Bashundhara R/A, Dhaka, Bangladesh. e-mail: mahady@iub.edu.bd

M. Rokonuzzaman
North South University, Plot 15 Block B, Bashundhara R/A, Dhaka, Bangladesh. e-mail: m.rokonuzzaman@northsouth.edu

1 Introduction

In today's data-driven world, data plays a significant role in almost every aspect of our lives. As more and more businesses rely on data analysis to drive decision-making, the importance of accurate and error-free data has become increasingly apparent (Hossen, & Sayeed, 2018). Large-scale data sets are collected and evaluated across various industries, and the problem of errors within those data sets has become more pressing. One common source of data for software applications is system logs, which often contain valuable information and error messages. However, many researchers and analysts overlook the importance of proper data cleaning and preparation before analysis (Hossen, & Sayeed, 2018).

This paper focuses on analyzing system logs to identify and address errors using machine learning techniques. The goal is to provide guidelines and solutions for optimizing software performance and reliability. By leveraging machine learning algorithms, this study aims to predict errors and provide solutions to prevent instability or inconsistency in the software during the development life cycle.

To achieve this goal, the proposed framework includes a search-based testing approach using deep neural networks. The framework incorporates strategies for code embedding, refactoring policy, mutation testing, and evaluating test cases. The system logs are preprocessed and analyzed using machine learning algorithms to identify patterns and predict errors. The resulting guidelines and solutions can be applied to any application logs, and the proposed data mining export code can be easily adapted for use in other settings.

Ultimately, the goal of this study is to contribute to the development of error-free software by providing a set of cleansed data for further investigation and analysis. By focusing on data quality and leveraging machine learning techniques, we hope to improve software performance, reliability, and efficiency.

The problem of the paper is related to the issues faced by companies due to the delivery of software without proper testing[15] and quality checking, resulting in increased development costs. Additionally, there are difficulties in creating unique or unfamiliar business transactions in the testing environment. Furthermore, even if the software is running well, errors in logs may arise due to not following proper standards or unknown process cycles. These errors may not stop the application's execution, but the author creates a considerable amount of logs in the production server, which affects performance and stability. Missing data integrity can also create errors and faults in transactions. Therefore, the paper aims to deal with these log issues and provide guidelines for their resolution without human intervention. The authors propose a unified algorithm for data cleansing, and the focus is on analyzing the logs of systems running in the production environment. The authors suggest machine learning algorithms to detect errors and propose solutions to optimize them to create an error-free application. The author aims to create a framework for analyzing big data to improve fault detection and problem identification.

The authors aim to develop a unified algorithm that can resolve data quality issues in unclean logs without the need for human intervention or master data (Al-janabi, & Janicki, 2016). The focus of their study is on analyzing the logs of running systems

in the production environment to provide guidelines and solutions for optimizing errors and creating error-free applications. They run machine learning algorithms to identify errors in the logs and suggest solutions to overcome them. The authors also emphasize the need for a framework for analyzing big data to improve fault detection and problem identification during data preprocessing (Hossen, & Sayeed, 2018).

2 Literature Review

In recent years, the use of machine learning for log analysis has gained significant attention in the field of software engineering (Mehra & Verma, 2019). Researchers have proposed various techniques and models for log analysis to improve software reliability, performance, and maintainability.

One approach is to use clustering algorithms to group log messages based on their similarity, which can help to identify common patterns and anomalies in the log data (Li et al., 2016). Another approach is to use classification algorithms to detect and categorize different types of log messages, such as errors, warnings, and informational messages (Xu et al., 2018).

Researchers have also explored the use of natural language processing (NLP) techniques for log analysis, such as topic modeling and sentiment analysis, to gain insights into the causes of log messages and to identify potential areas for improvement (Zhou et al., 2019).

In addition to machine learning, researchers have also proposed other approaches for log analysis, such as rule-based systems and pattern-matching techniques (Mehra & Verma, 2019). Rule-based systems use predefined rules to detect specific types of log messages, while pattern-matching techniques search for specific patterns or sequences of log messages that may indicate a problem.

Despite the various approaches proposed for log analysis, there are still challenges and limitations in this field. One challenge is the complexity and variability of log data, which can make it difficult to develop accurate models and algorithms (Li et al., 2016). Another challenge is the lack of labeled data for training and testing machine learning models, which can limit their effectiveness (Xu et al., 2018).

Despite these challenges, the potential benefits of log analysis using machine learning are significant, including improved software quality, reduced downtime, and increased productivity (Zhou et al., 2019).

3 Research Design

The aim of this research is to detect and classify system errors by using a Random Forest feature selection algorithm, to minimize the error rate and predict possible error solutions based on current errors.

3.1 Data Collection

The system log files will be collected from a live-running application that is privately available. The data will be selected from a specific time period to ensure consistency in the data. The dataset will be cleaned using the random sampling technique to reduce the size of the dataset and make it more manageable. Those applications are creating a huge number of log files (Al-janabi, & Janicki, 2016) where authors have seen many error tags. Since the logs are huge and there is a lack of a big data handling machine. The author picked up 1 week of data from 2021, which are about 921MB,

3.2 Data Pre-processing

Data pre-processing is an essential step to ensure data quality and prepare the dataset for analysis. The data will be translated into a structured format and validated to handle missing values correctly. Data cleaning techniques will be used to remove null values and incomplete data. To achieve high accuracy, it is also necessary to handle missing values correctly (Hossen, & Sayeed, 2018). To programmatically prepare those files, one must first convert the raw log file to a CSV file (DIMOV & OROZOVA, 2020). An algorithm has been developed to mine the dataset and extract only error logs.

Example of a Computer Program in PHP:

```
Begin

$finalFile = fopen( 'file.csv', 'a');
$GetData = array_map( 'str_getcsv', file('file.csv'));

for ($n=0;$n<=count ($GetData); $n++)
{
$EechLineData = $GetDatal[$n];
$ExplodeLog = explode( 'delimiter',$EechLineData);
$Explodeagain = explode( 'basedOnNeed',$Explodelog);
    if ($Explodeagain=="Error")
    {
        //run validation process

        //check for meaning full data

        $a = 'CleanData';

        fputcsv($finalFile, $a, delimiter: ™,");
   }
}

End.
```

3.3 Feature Selection

Feature selection is a crucial step in the data mining process, which involves identifying and selecting the most relevant features for analysis. Random Forest feature selection will be used to select the most relevant features for analysis. The algorithm will be trained on the dataset, and the most important features will be selected for analysis.

3.4 Error Detection and Classification

The selected features will be used to detect and classify system errors. The algorithm will be trained to predict possible error solutions based on current errors. The error logs will be categorized into groups based on the feature selection method to identify patterns and relationships between errors.

3.5 Evaluation

The performance of the algorithm will be evaluated based on accuracy and precision. The results will be compared to existing methods for detecting and classifying system errors. The data was not organized because it was gathered in its raw form, from sources. It contains null values, incomplete data, some missing values, and a sampling date format that is inconsistent. Thus, data analysis had to be performed using any relevant language and algorithm to check the dirtiness of the data, and then a well-structured dataset could be generated using data cleaning techniques, algorithms, and procedures, which could then be used for analysis or visualization (Kumar, & Khosla, 2018)

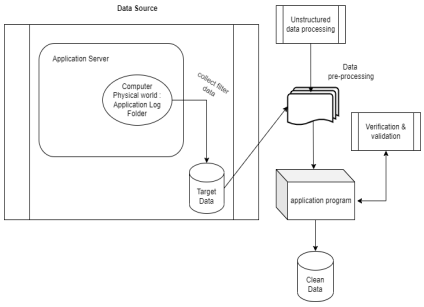

Fig. 1 Data analysis model.

The research will provide insights into the causes of system errors and offer possible solutions to minimize the error rate. The Random Forest feature selection algorithm will be useful in identifying the most relevant features for analysis, and the error classification system will provide a framework for predicting possible error solutions. The results of this research will contribute to the field of data mining and error analysis.

The study will only consider a specific time period and may not provide a comprehensive analysis of system errors. The dataset is based on a live-running application, and the results may not be generalizable to other applications. The study may also face computational limitations due to the large size of the dataset.

4 FINDINGS

To achieve a clean, error-free data set for analysis in machine learning, pre-processing raw data is a critical step, as outlined by Haider, Zhao, and Meran (2020). In this study, customized algorithms were employed to clean the data, and Python or R are recommended for this task due to their built-in libraries for statistical analysis and interpretation, as noted by Hossen and Sayeed (2018). The data analyzed in this study covers a one-week period, during which a total error count of 4042852 was observed.

Upon analysis of the cleaned data set, it was found that only warning and notice-type errors occurred, which did not cause the application to stop executing any script. These types of errors are non-fatal and do not halt script execution. The cleaned data set contains various columns related to the data, such as Title (log type), Type (error type), the affected variable and line number, error, and filename. Table I provides an overview of our cleaned data.

Table 1 Mined Data Set

Title	Type	Variable,Line	File
ERROR	Notice	payment, 192	/controller/./billing.php
ERROR	Notice	courses, 119	/controller/./billing.php
ERROR	Notice	semester,120	/controller/./billing.php
ERROR	Warning	LastName,63	/controller/./landing.php

Table II illustrates the error frequency which has been generated using Anaconda a big data handling tool. The first column contains information about the error type, then the second column represents the error file location and the third column shows how many errors occurred on those files.

According to the information author gathered, the following errors were discovered, mentioned in Table III:

Table 2 Error type and Frequency

Error Type	***Occurred***	***Frequency***
Notice	controller and view files	36,40,423
Warning	View file	3,99,861

Table 3 Most occurred error

1	Invalid argument supplied for foreach()
2	Trying to get property of non-object
3	Undefined property

To represent the analyzed data insight visualization software such as Tableau, Power BI, or Rapid Miner could be used (Kumar, & Khosla, 2018). As the author is familiar with Rapid Miner, it has been used to analyze data.

Table 4 Occurred Errors

Errors	***Frequency***	***Fraction***
Undefined property	2494821	0.62
get property of non-object	1038676	0.21
Invalid argument foreach()	358041	0.09

Table IV contains information about different types of errors and their absolute value fraction in a system. There are three types of errors listed in the table: ”Undefined property,” ”get property of non-object,” and ”Invalid argument foreach().” For each type of error, the table provides the number of occurrences (absolute value) and the fraction of their occurrence in the system. The total number of errors listed in the table is 3, and the total number of occurrences is 4,005,538.

Table 5 Most occurred error

Title	***Errors Type***
ERROR	Notice
ERROR	Notice
ERROR	Warning
INFO	Some Text
INFO	Some Text
ERROR	Warning
ERROR	Notice

Table 5 is used to detect system total error and non-error count which has shown in Figure 2.

Raw data is often incomplete, inconsistent, and redundant, making it unsuitable for direct data mining. Therefore, advanced analysis techniques are required to process

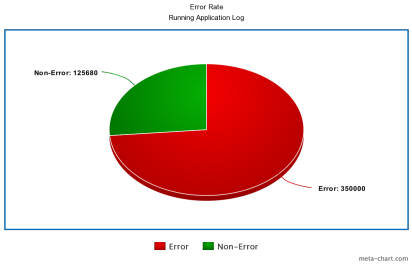

Fig. 2 Error rate

the data (Haider, Zhao, & Meran, 2020). In this study, the author decided to use one day of acquired data and apply machine-learning techniques (Hossen & Sayeed (2018). The data had a shape of (475680, 5), but since the CSV file did not provide numeric data, a method for transforming nominal data into numeric features was used (Zdravevski, Lameski, Kulakov, & Kalajdziski, 2015). To detect whether each row of the data represents an error or not, the raw data had to be converted into a numerical format (Hong, Ashwin, Johnson, & Xiaofei, 2016). Excel was used to generate the desired pattern, which can be seen more clearly in Figure 3.

Table 6 Error Pattern Mining

Log Tag	***Type***	–	–	–
ERROR	Notice	1	1	1
INFO	Text	0	4	0
INFO	Text	0	4	0
ERROR	Notice	1	1	1
ERROR	Warning	1	2	1
ERROR	Warning	1	2	1

The inclusion of the affected file name and line number in the cleaned data set mentioned in Table I is expected to be beneficial for future research. With this information, an AI system can locate faulty files and affected variables mentioned there, and suggest solutions for the identified issues. In addition, a new approach called Deep Check can be proposed for testing Deep Neural Networks (DNN) using symbolic execution and program analysis. Deep Check uses a white-box technique to facilitate symbolic execution and identify critical elements in the DNN. (Wardat, Le, & Rajan, 2021).

In order to utilize the data for analysis, it was transformed into a structured format. To ensure its reliability, the information was verified and checked for any instances of missing data. (Hossen & Sayeed, 2018).

Our proposed methodology requires the use of supervised learning, with logistic regression being the preferred model due to our data's pattern. Given the large input data, it is more convenient to create a prediction model. To ensure the accuracy of

the preprocessing, mining patterns, and analysis, we selected a single day's data to train and test the logistic regression model. The system must differentiate between errors and non-errors in the selected data. We split the data into training and test sets, with 20% used for testing and 80% for training, resulting in a total dataset of (475680, 2), with (380544, 2) for training and (95136, 2) for testing.

4.1 Model Training

In order to proceed with the training phase, it is important to select a suitable data source and identify key features that are critical to the process. The accuracy of the trained model must then be evaluated to ensure that it produces reliable results. Subsequently, the ML model can be utilized to make predictions based on data analysis. (Hossen, & Sayeed, 2018).

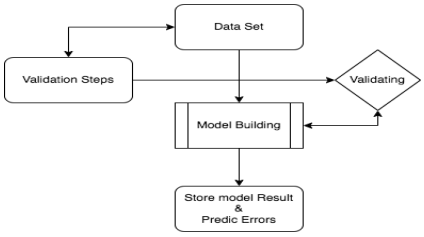

Fig. 3 Training Process

It is suggested to use a technique such as oversampling or undersampling to balance the distribution of the imbalanced data during model training. Oversampling involves increasing the number of instances of the minority class, while undersampling involves decreasing the number of instances of the majority class. Both techniques can help to ensure that important information is not lost during the elimination of examples. Additionally, the authors wanted to consider using a more sophisticated sampling technique such as SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling) to create synthetic instances of the minority class, which can help to improve the balance of the dataset. By incorporating these techniques, the authors can improve the quality of their model and ensure that it produces accurate results. (Gao, Khoshgoftaar, & Napolitano, 2012).

4.2 Model Evaluation

Precise information is a crucial factor in information analytic execution due to location and anticipation. The author observed a training data accuracy of 0.83 after removing the complete out layer, indicating the accuracy of their hypothesis. The

high accuracy can be attributed to the thorough preprocessing and mining of the data sets. Eliminating the outer layers resulted in a higher accuracy of 87 percent in clean data sets. The Random Forest classifier was used to train and construct a model to recognize data quality from the dataset (Hossen, & Sayeed, 2018). After the removal of the entire outlier and the evaluation of Random Forest, the application achieved an accuracy of 83 percent. This high accuracy was possible due to the high quality of the processed data. Therefore, the conclusion is drawn that to achieve high accuracy, reliable data is necessary. A distributed algorithm like mining outliers can be used to make data sets more reliable (Liu, Guo, & Sun, 2017).

The results of the ML analysis showed that errors in the log file can indicate errors in the system. Therefore, by predicting errors in the log files, the author can detect errors in the system.

5 PROPOSED MODEL

The proposed solution suggested by the authors includes building a data-cleaning function using machine learning algorithms to predict potential errors in data. To further improve the application's quality, the authors suggest creating loosely coupled modules and classes with global variable declarations and high cohesion. The actual code-writing process should involve array or variable declarations, type declarations of variables, isset checks, empty value checks, and type checks before using any variables. Finally, a Unit test is recommended to optimize unwanted errors.

To incorporate this proposed solution, developers can follow these guidelines during the development process of an application or feature to reduce the chances of errors and bugs. By creating loosely coupled modules and classes with global variable declarations, developers can make sure that their code is easy to maintain and update. Moreover, by following the suggested coding practices, such as array or variable declarations, type declarations of variables, isset checks, empty value checks, and type checks before using any variables, developers can ensure that their code is error-free and more reliable. Finally, by conducting a Unit test, developers can identify and optimize any unwanted errors, further improving the application's overall quality.

Overall, incorporating these guidelines into the development process can optimize the production cost and time, make the application scalable, reliable, and faster, and ultimately, ensure an error-free log or output.

6 CONCLUSIONS

The purpose of this study was to create a data cleaning function that can improve data quality by identifying and predicting errors. The research question focused on whether machine learning techniques can be used to improve data quality in software

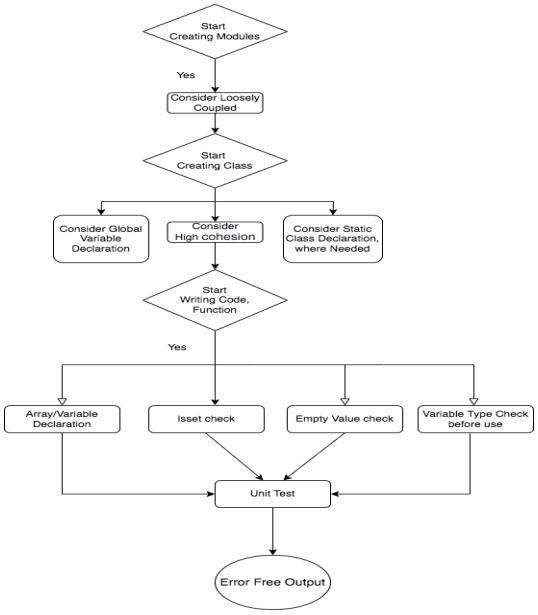

Fig. 4 Proposed Model.

applications. The literature review highlighted the importance of data cleaning in machine learning and the use of various techniques such as classification models to identify and correct errors in software applications.

The study analyzed one week's worth of data and found a total of 4042852 errors, which were mainly warning and notice-type errors. The data cleaning function developed by the authors was able to identify and correct these errors, leading to improved data quality. The authors also provided guidelines for developing error-free software applications, including loosely coupled module creation, global variable declaration, and unit testing.

Future work could focus on expanding the scope of the study to include data from multiple sources and different time periods. The authors could also explore the use of other machine learning techniques, such as clustering, to identify and correct errors in software applications. Also, more data structuring recommendations will be included. Integrity constraints (IC), such as Functional Dependencies (FD), can be used in conjunction with Machine Learning to classify the type of error to be captured in the event of a data set with an inaccurate value (Hossen, & Sayeed, 2018). To fully appreciate the data's potential for bringing significant benefits to a variety of businesses, it is necessary to learn from it (Haider, Zhao, & Meran, 2020). Some "Context of source code processing" will be considered where a mutation in the context plays as refactoring source code and 1-Time, K-Time mutation will be played an important role in the concept (Maryam, Zhuo, Lei, Hadi. 2021). In that study, the author will focus on a few research questions, and the author will try to solve them one by one by considering best practices.

In conclusion, this study demonstrated the importance of data cleaning in machine learning and provided a solution for improving data quality in software applications. By using machine learning techniques and following the guidelines provided by the authors, software developers can create error-free and efficient applications that are scalable, reliable, and faster. Future research in this area could lead to further improvements in data quality and software development practices.

References

1. Hossen, J., & Sayeed, S. (2018, September). Modifying cleaning method in big data analytics process using random forest classifier. In 2018 7th (ICCCE) (pp. 208-213). IEEE.
2. Al-janabi, S., & Janicki, R. (2016, July). A density-based data cleaning approach for deduplication with data consistency and accuracy. In 2016 SAI Computing Conference (SAI) (pp. 492-501). IEEE.
3. Mehra, S., & Verma, R. (2019). An approach towards log analysis of large-scale systems. International Journal of Computer Applications, 181(41), 18-22.
4. Li, Q., Zhao, C., & He, X. (2016). A clustering-based log analysis approach for improving software reliability. Journal of Systems and Software, 118, 197-212.
5. Xu, X., Li, Y., Wang, Y., Li, X.,& Li, B. (2018). Log classification based on multi-view feature learning. Information and Software Technology, 98, 126-139.
6. Zhou, M., Zhang, Z., Li, Y., & Zhang, H. (2019). Log analysis using natural language processing techniques: A survey. Journal of Systems and Software, 151, 99-115.
7. DIMOV, T., & OROZOVA, D. (2020, June). Software for Data Cleaning and Forecasting. In 2020 21st International Symposium on Electrical Apparatus & Technologies (SIELA) (pp. 1-4). IEEE.
8. Kumar, V., & Khosla, C. (2018, January). Data Cleaning-A thorough analysis and survey on unstructured data. In 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 305-309). IEEE.
9. Wardat, M., Le, W., & Rajan, H. (2021, May). DeepLocalize: fault localization for deep neural networks. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 251-262). IEEE.
10. Gao, K., Khoshgoftaar, T. M., & Napolitano, A. (2012, December). A hybrid approach to coping with high dimensionality and class imbalance for software defect prediction. In 2012 11th international conference on machine learning and applications (Vol. 2, pp. 281-288). IEEE
11. Haider, S. N., Zhao, Q., & Meran, B. K. (2020, July). Automated data cleaning for data centers: A case study. In 2020 39th Chinese Control Conference (CCC) (pp. 3227-3232). IEEE.
12. Maryam V.Pour, Zhuo Li, Lei Ma, Hadi Hemmati. 2021. A Search-Based Testing Framework for Deep Neural Networks of Source Code Embedding. Calgary, Canada. DOI: https://doi.org/10.1145/1188913.1188915.
13. Zdravevski, E., Lameski, P., Kulakov, A., & Kalajdziski, S. (2015, September). Transformation of nominal features into numeric in supervised multi-class problems based on the weight of evidence parameter. In 2015 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 169-179). IEEE.
14. Hong Liu, Ashwin K. TK, Johnson P Thomas and Xiaofei Hou. 2016. Cleaning Framework for BigData.
15. Hasan, M. T., Mahal, S. N., Bakar, N. M. A., Hasan, M. M., Islam, N., Sadia, F., Hasan, M. (2022). An Unified Testing Process Framework for Small and Medium Size Tech Enterprise with Incorporation of CMMI-SVC to Improve Maturity of Software Testing Process. In ENASE (pp. 327-334).