INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

RESEARCH PROSPECTS FOR APPLYING DATA MINING METHODS IN SOFTWARE ENGINEERING LIFECYCLE

S. Sucithra
Student,
Department of Computer Science,
Rajeswari College of Arts and Science for Women,
Villupuram, Tamil Nadu, India.

Abstract

Software industry is a field of study humanized in the software engineering. Every day, people come up with new ways to do things and new frameworks. For software engineering to continue, it must be able to adapt and work with things. Software engineering uses natural language processing, data analysis, machine learning, and intelligence. Since people want software, we need to conduct a lot of research on software development data. It's really challenging to handle a large quantity of data without being able to process it and look at the amount of data. Data mining methods are used to enhance the creation of software. This study examines the way of text mining, clustering, and classification techniques are used. This shows what these data mining methods can do help enhance the life cycle of software development. Data mining methods include important for improving the efficiency of software development and effective. The paper talks about the uses and results of classification, clustering, and text mining methods, in data mining.

Keywords: Software engineering, text mining, SDLC, data mining, clustering, and classification.

1. Introduction

The software industry is a field that really a important for the software industry. It is always changing because new methods and algorithms are added. This implies that the fundamental methods of performing tasks in software engineering must undergo change. The software engineering field must adapt to research areas. There is a problem. There is much data available now, and the systems we use to make software are really complicated. This causes many errors. Costs a lot of money.

The software engineering field is still evolving. This is a significant challenge for the software industry and software engineering field. These problems need to be addressed. Therefore, software engineering must change. It must include ideas from areas such as artificial intelligence, big data, machine learning, and data mining.

Data mining is important in this context. It helps in software development at many stages. It provides information that makes things more efficient and helps us

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

make good decisions. Data mining is crucial for this purpose. This is more helpful than in some areas of research. Software engineering and data mining are key, to making things better.

This study shows us the bad things about using data mining in software development. This document is classified into five sections. Section I is the outlines of the purpose a study. Section II is important because it discusses the usefulness of software development through text mining.It tells us the way of text mining helps with security analysis by finding bugs in versions, creating test cases, and making changes to versions. Section III presents the testing's outcomes methods that use mining techniques. These methods are important. Section IV discusses how clustering and classification algorithms affect parts of the software life cycle. The study concludes by examining all these ideas. This shows that we need to continue research to fix the problems we have now and to make mining methods work better with software engineering. Mining methods must be made a part of software engineering.

Text Mining Techniques in Development Phases:

The thesis examined the use of text classification techniques for software requirements that are used in companies and their applications. It attempted various methods to determine the most effective approach. The main goal of the thesis was to see how well the classifier could figure out what kind of requirement each statement was, every time it looked at one. A study introduced an arrangement to find configuration errors in a business software system. The researchers examined bug reports and used this information to create a new model. This model helps determine the type of bug. This could be due to a configuration bug or some other issue. If it is a configuration bug, the development team fixes it. They used methods to classify the bugs and select the important features. The model is based on a summary of the language used in bug reports, as shown in Figure 1. The configuration bugs are. The development team then addresses the configuration bugs. In this way, we can sort bugs into configuration bugs or other types of bugs. The development team addresses bugs that are identified as configuration bugs.

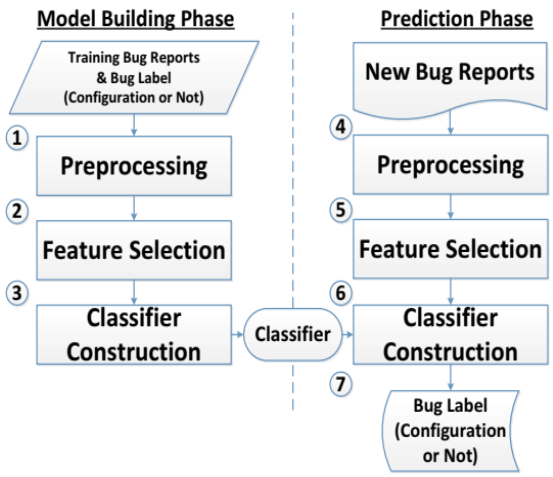

Fig 1.The suggested approach for predicting bugs

To stop the software from failing and to make the work easier, our algorithm finds the bug label. Tell the team about it. Someone

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

came up with A method was developed to identify security bugs in software by analyzing the text. This is important for people who use the software and for those who create it. Bugs that are related to security are usually called security-related bugs (SRB).

Sometimes, people make mistakes and call them something else, such as BSRB. When this happens, it can cause problems for the entire software system. The software can be messed up because of these mistakes. Our algorithm attempts to identify these Security-Related Bugs so that the team can fix them and improve the software. The incorrectly labeled security bugs were fixed. The labels are now sorted out in a way that understands what people mean when they write things. This is done by looking at the words people use, which is called text mining. The security bugs are labeled correctly now.

The Historical Information for Smell detection method also known as HIST uses information about how things changed over time to find problems in code. It looks for five problems: Blob, Parallel Inheritance, Feature envy, shotgun surgery, and divergent change. Historical Information for Smell detection was tested in two studies. These studies used code projects and different ways to analyze the code.

The results showed that Historical Information to Smell detection was able to find problems between 58% and 100% of the time. However, when it came to being precise, Historical Information for Smell detection was correct between 72% and 86% of the time.

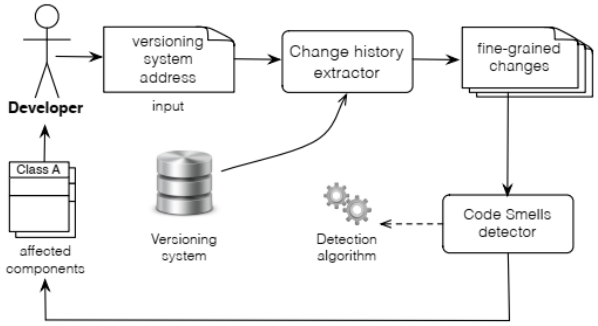

Fig 2. HITS model

Shah and Pfahl conducted research on mapping to improve software development. They examined software development data from a few places, such as the things that people wrote in natural language of the remarks in a design documents' source code and demands for changes to be made to the software.

Software Testing Mining Techniques

The mining data is effective in finding patterns and things that are similar in test cases. It can also be applied to lower of the quantity of necessary test instances. The main thing that the reduction of test suites by the data mining is to look at what you put into the software and what you get out of it. This helps to get rid of test cases that are not needed or that do not work.

Data mining techniques are great because they are smart and they get the job done quickly. They can also be difficult to understand. The thing is, how well data mining works depends on how examples you have to teach it with.

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

Using data mining approaches, test suites were reduced and divided into three primary categories:

- Classification
- Cluster Analysis
- Mining Frequent Itemset (MFI)

Each category has one or more subcategories. Figure 3 shows the proposed classification.

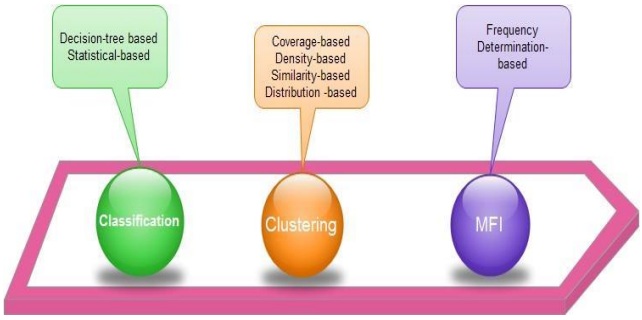

Fig 3.Testing methods for mining

Khan and his team proposed a way to group things to reduce or get rid of extra test documents. We do not have tests to begin with; therefore, it is very important to make the test suite smaller when testing software. The test suite we have is very large. It has many extra test cases, as the people who made it explained. The test suite is too large because it has many test cases; thus test suite minimization is necessary for software testing, and the k-means clustering technique that Khan and his team proposed can help with that. The test suite size can be a large because of these factors. This includes the fact that program variables can have many values. Also when the requirements of a program change people usually add test cases to the test suite.

The four groups in a test cases, which are called C1, C2, C3 and C4. Using this method, one test case can be selected from each test case group. This means that a test suite with four test cases can be created, which is a relatively small test suite. We were able to lower the quantity of test cases.

With 100% call coverage by approximately 80%. This is because we used our prediction techniques. The prediction technique helped us lower the quantity of test cases. We had many test cases. Our prediction technique helped minimize these issues. Prediction technique to make this happen to the test cases, with 100% call coverage.

Researchers used data mining techniques, such as clustering and classification, to find ways to reduce the number of test cases. Data mining techniques, such as clustering and classification, have been helpful. Some studies, such as those in references 12 and 13, showed that the number of test cases can be reduced. This made the testing process faster and more efficient than the previous version. It also saves money and time by reducing the number of test cases that must be repeated. Data mining techniques, like clustering and classification made a difference in this area.

Paper looked at test data to find patterns using a way of searching through data. The first step was to determine whether the test parameters actually mattered to the results. They then cleaned the data, removed the bad data, and spread out the remaining

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

data to take a closer look. Chantrapornchai et al. [14] introduced a method to reduce the number of test cases for black-box and white-box testing using K-means clustering algorithms. They used a called code coverage to analyze the variables and reduce the number of variables before clustering. The code coverage helped to lower the number of variables.

Chantana Chantrapornchai and others reduced the test cases from 648 to 486. Then to 324 for K-means black box testing.The K-means clustering algorithms maintained the code coverage for the test cases.The test case reduction method used by Chantana Chantrapornchai and others is useful, for K-means black box testing and K-means white box testing.

Clustering and classification used in Software Engineering:

Dulal Chandra Sahana gives us some useful information about software defect prediction mining algorithms. He looks at a lot of mining algorithms to see which ones are good, at predicting defects. He found that the Naive Bayes and Logistic algorithms performed the best overall. Dulal Chandra Sahana tells us that these two algorithms have the performance when it comes to predicting software defects.

The people who did this study only looked at a ways to classify things. They compared Decision Tree, Bayesian, Rule-Based and Logistic Regression methods. Some organizations, including the MGF, used information from the NASA MDP library to train their systems. This library contains data that can be used by everyone. The NASA MDP repository has 12 sets of data that can be used for studies. They have these 12 datasets in the NASA MDP repository.

Table 1: Accuracy among the Classifies

Methods	NB	LOG	DT	JRip	OneR	PART	J48	J48G
CM1	83.94	87.68	**89.13**	86.23	**89.13**	73.91	86.23	86.96
JM1	81.28	**82.02**	81.57	81.42	79.67	81.13	79.8	79.83
KC1	83.05	**86.87**	84.84	84.84	83.29	83.89	85.56	85.56
KC3	77.5	71.25	75	76.25	71.25	81.25	80	**82.5**
MC1	**94.34**	99.27	99.25	99.22	99.3	99.19	99.3	99.3
MC2	66	66.67	56.86	56.86	56.86	**70.59**	52.94	54.9
MW1	79.25	77.36	85.85	86.79	85.85	**88.68**	85.85	85.85
PC1	88.82	92.11	**92.43**	89.14	91.45	89.8	87.83	88.49
PC2	94.29	99.05	**99.37**	99.21	**99.37**	**99.37**	98.9	98.9
PC3	34.38	**84.67**	80.22	82.89	82.89	82.67	82.22	83.56
PC4	87.14	**91.79**	90.18	90.36	90.18	88.21	88.21	88.93
PC5	96.56	96.93	97.01	**97.28**	96.9	96.93	97.13	97.16

Ashish Kumar and others try to find software design patterns by using software metrics and techniques that involve classification (Fig 4). Our research was conducted in two parts. First, we collected data based on metrics. We then attempted to identify software design patterns and used software metrics to create datasets and train classifiers.

We propose to find design patterns in software using machine learning methods. These methods analyze datasets based on metrics. We used software metrics and machine learning to identify software design patterns. The new method was tried out with three open-source software programs: JHotDraw, QuickUML, and J Unit. The results

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

of JHotDraw, QuickUML, and JUnit were then examined closely.

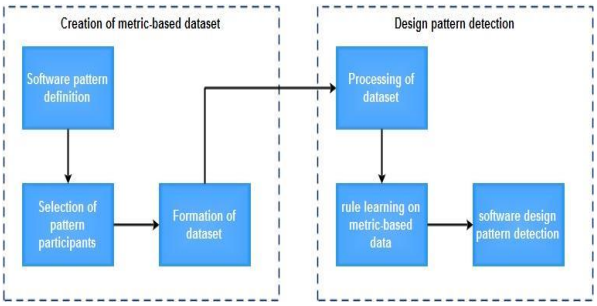

Fig 4 Model of Software Design Pattern

One problem with their approach is that they did not get to know every type of design pattern that appears in the software code. When they were trying to determine the pattern of something, the people who wrote this method looked at classes of design patterns. They should have looked at design patterns to perform the job. The method they used had a problem because it did not look at every class of design patterns that can be found in software source code.

Shin and Williams looked at how they could use code-churn measurements and complexity in fault-prediction models to predict vulnerabilities. They conducted a study on Mozilla Firefox to determine whether this would work. They used many metrics in their study. There are 18 quality metrics, five code churn metrics, and a statistic about past faults. They wanted to determine whether these metrics could help predict vulnerabilities in Mozilla Firefox and attempted to classify the files to determine which files are defective and vulnerable.

The people who did this work said that all the methods they tried yielded similar results. The models that predict faults and vulnerabilities in files performed similarly in terms of predicting vulnerabilities. They were approximately 83 percent of the time and accurate approximately 11 percent of the time. This is interesting because defective source code files are much more common than source code files. It was approximately seven times more common. The fault and vulnerability prediction models and the vulnerability prediction models are really the thing and they both do a pretty good job of finding defective and vulnerable files, specifically vulnerability prediction and fault and vulnerability prediction models. The authors looked at the results. They believe that models that predict faults and are based on the usual measurements can also be used to predict vulnerabilities.

The authors believe that more work needs to be done on these fault prediction models to improve their ability to predict vulnerabilities. This means that the models should be good at identifying vulnerabilities and not providing too many false warnings. The models should also be able to find most of the vulnerabilities that're there which is what the authors mean by a high recall, for vulnerability prediction especially for the fault prediction models.

The Naïve Bayes classifier was used by Zhu et al. To develop a design for the software process model. They used the Naïve Bayes classifier method to make models, from the software development logs.

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

The Naïve Bayes classifier method does a things. First, it provides a label for each action performed. Then it uses something called k-means clustering algorithms to get the meaning of the things.

The Naïve Bayes classifier approach does this in two steps to get a record of what is done. In this step, two types of learning are used: one where it is told what to do and one where it figures things out on its own. This study aims to find a way to calculate harmonic averages to determine the accuracy and performance of a classifier in remembering things.

It also discusses a method for determining the number of groups when we do not know what we are looking for by using something called second derivatives. To be fair, the people conducting this study checked how well a classifier worked. This can be seen in a paper called.

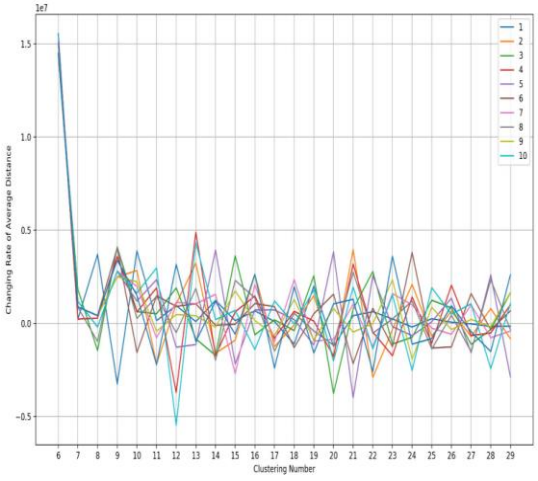

Fig5.Selecting the Number of Clusters

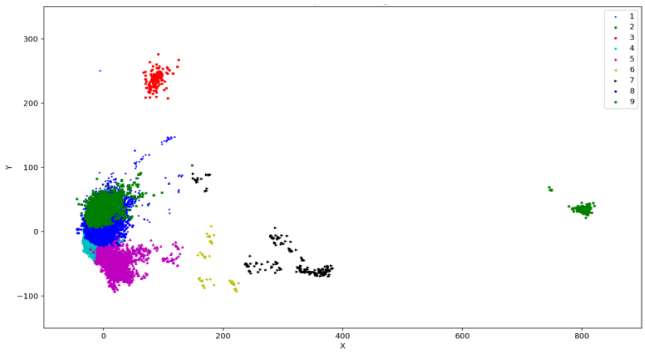

Fig 6.Effect of optimal cluster number

To undertake knowledge discovery, Chandrasekaran et al. [20] investigated the comparative results of applying five mining algorithms to combination testing. The findings of the experiment specifically show that data mining algorithms act similarly to generic software applications. This implies that data mining techniques may benefit from the application of CT. Table 2 and Figures 7 and 8 show the comparison results for the two models.

Table 2: Branch Coverage Results of T-way Testing

Test set	Apriori	EM	J48	Simple K-Means	LibSVM
Default Configuration	28.79%	37.64%	36.64%	21.89%	34.96%
Negative	66.03%	50.54%	55.32%	66.24%	28.47%
1-way	55.52%	52.99%	52.73%	59.51%	24.47%
2-way	66.55%	53.80%	54.60%	69.53%	43.77%
3-way	68.62%	53.94%	59.77%	70.39%	54.63%
4-way	68.62%	53.94%	59.77%	70.39%	54.80%
5-way	68.62%	54.08%	59.77%	70.39%	54.80%
6-way	68.62%	54.08%	59.77%	70.39%	54.89%
6-way &Negative	68.97%	55.30%	59.77%	70.67%	55.34%

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

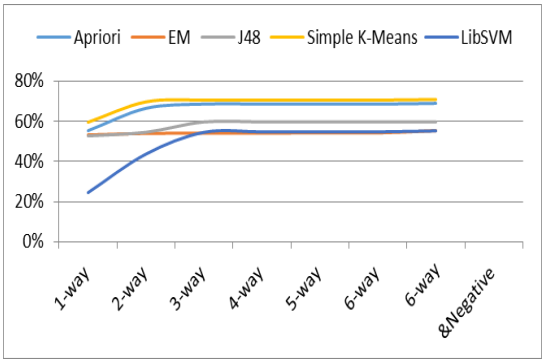

Fig 7.Growth of Branch Coverage

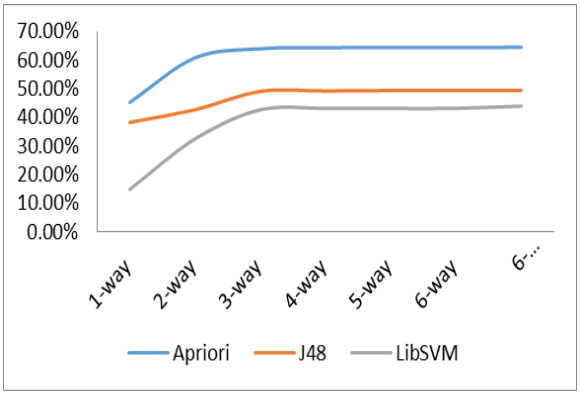

Fig 8.Growth of Mutation Coverage

Table 3: Mutation Coverage Results of T-way Testing

Test set	Apriori	J48	LibSVM
Default Configuration	31.53%	29.66%	26.11%
Negative	59.90%	40.39%	19.46%
1-way	45.27%	38.05%	14.99%
2-way	60.93%	42.56%	32.61%
3-way	64.10%	49.05%	42.72%
4-way	64.47%	49.19%	43.19%
5-way	64.50%	49.35%	43.19%
6-way	64.53%	49.38%	43.21%
6-way&Negative	64.63%	49.38%	44.02%

Conclusion

Data mining is really important in software engineering these days. People are doing lots of research on this. They have found many problems that need to be solved.They have also come up with questions that need to be answered. This study is about how data mining affects the process of making software. Data mining techniques are very helpful in making the software development process better and cheaper.They also help reduce the time it takes to make software and the costs that keep coming up.This study talks about the problems that researchers have found and what they have learned from using data mining techniques like text mining, clustering and classification in software engineering.Data mining techniques, like text mining and classification are used a lot in software engineering to make things better. This study will be very helpful to researchers in matching the results and architecture of various proposed studies and sticking with their research.

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

References

1. Dawid Zima, Modern Methods of Software Development, Task Quarterly 19, No 4, pp. 481–493, 2015.
2. Max Bramer, Principles of Data Mining, Second Edition, Springer, 2013.
3. JapaSwadia, A Study of Text Mining Framework Automated Classification of Software Requirement sin Enterprise Systems, Master thesis, Arizona State University, May 2016.
4. X. Xia,Automated Configuration Bug Report Prediction Using Text Mining, IEEE 38th Annual International Computers, Software and Applications Conference, 2014.
5. M. Gegick, P. Rotella, and T. Xie, "Identifying security bug reports via text mining: An industrial case study," 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), Cape Town, 2010, pp. 11-20.
6. F. Palomba, G. Bavota, M. D. Penta, R. Oliveto, D. Poshyvanyk and A. De Lucia, "Mining Version Histories for Detecting Code Smells," in IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 462-489, 1 May 2015.
7. Shah, Faiz A., & Pfahl, D. "Evaluating and Improving Software Quality Using Text Analysis Techniques - A Mapping Study." REFSQ Workshops (2016).
8. A. Saifan, Ahmad, EmadAlsukhni, HanadiAlawneh, AyatAL_Sbaih,"Test case reduction using data mining technique." International Journal of Software Innovation (IJSI) 4.4 (2016): 56-70.
9. Raamesh, Lilly, and G. V. Uma, "Reliable mining of automaticallygenerated test cases from software requirements specification(SRS)." arXiv preprint arXiv:1002.1199 (2010).
10. Keyvanpour, Mohammad Reza, HajarHomayouni, and HosseinShirazee, "Automatic software test case generation: An analytical classification framework." International Journal of Software Engineering and Its Applications 6.4 (2012): 1-16
11. Khan, Fayaz Ahmad et al. "An Efficient Approach to Test Suite Minimization for 100% Decision Coverage Criteria using K-Means Clustering Approach." (2015).
12. Ilkhani, Ali, &Abaei, Golnoush. (2010). Extraction test cases using data mining reduce testing costs. 620 - 625.
13. B.Subashini,D.JeyaMala, Reduction of Test Cases Using Clustering Technique, International Journal of Innovative Research in Science, Engineering and Technology Volume 3, Special Issue 3, March 2014
14. H. Wang, L. Bai, M. Jiezhang, J. Zhang and Q. Li, "Software Testing Data Analysis Based on Data Mining," 2017 4th International Conference on Information Science and Control Engineering (ICISCE), Changsha, 2017, pp. 682-687.
15. Chantrapornchai &Kinputtan, K. &Santibowanwing, A.. (2014). Test case

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE
Scholorly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
Since 2012
www.ijcsjournal.com Volume 14, Issue 1, No 03 2026 ISSN: 2348-6600
REFERENCE ID: IJCS-586 PAGE NO: 084-093

reduction case study for white-box k-boblack-box using data mining. International Journal of Software Engineering and Applications. 8. 319-338. 10.14257/ijseia.2014.8.6.25.

16. Dulal Chandra Sahana, Software Defect Prediction Based on Classification Rule Mining, Master's thesis, National Institute of Technology Rourkela, 2013.
17. Dwivedi, Ashish, and Anand Tirkey, (2018). Software design pattern mining using classification-bases techniques. Front. Comput. Sci.
18. Shin, Y. Wil;liams, L. 2013. Can traditional fault prediction models be used for vulnerability prediction? Empir.Softw. Eng. 18, 1 (2013), 25–59.
19. RUI ZHU et al, Automatic Real-Time Mining Software Process Activities From SVN Logs Using a Naive Bayes Classifier, IEEE Access, October 21, 2019.
20. JaganmohanChandrasekaran et al.[], Applying Combination Testing to Data Mining Algorithms, 10th IEEE International Conference on Software Testing, Verification and Validation Workshops.