MANTECH Publications

Journal of Data Mining and Knowledge Engineering
Volume 7, Issue 1, January-June 2022

Protecting Users' Privacy While Mining and Profiling Their Code

Komal Rawat
Professor
Department of Computer Science & Engineering
Nanhi Pari Seemant Institute of Technology
Corresponding Author's Email: - *krawat22@yahoo.com*

Abstract

Privacy-preserving data mining is a critical topic in data mining today. Because numerous companies and individuals are producing sensitive data or information these days, they do not wish to divulge their sensitive data, yet that data might be beneficial for data mining. As a result of privacy-preserving mining, such data may be mined usefully without jeopardizing its privacy. Because the data is now safe owing to encryption, privacy may be retained by encrypting the database that is to be mined. Code profiling is a branch of software engineering in which we may utilize data mining to identify knowledge that will be valuable in future software development. We used privacy-preserving mining of code profiling data such as software metrics of diverse codes in this work. The accuracy of data mining results on real and encrypted data is compared. We also looked at the outcomes of privacy-preserving mining in code profiling data and discovered some intriguing results.

Keywords: *Data mining, Privacy protection, code profiling, and correlation coefficient*

INTRODUCTION

Software systems are typically complicated and difficult to grasp. This complexity, when combined with extensive dependencies and disparate programming styles, delays development and maintenance operations, leads to errors and failures, and ultimately raises software costs. Software engineering operations create a massive quantity of data, which, when correctly exploited using data mining techniques, may assist

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

give insight into many aspects of software development processes. Because of the massive volume of personal data created daily, privacy-preserving data mining is critical. The preservation of privacy is a critical concern in the realm of data mining. Data sharing benefits several applications, particularly data statistics and data mining. However, shared data may contain private information about the data's owner. It poses a considerable danger of compromising the privacy of the data owner [2]. Most of the time, people are unwilling to give their personal information, so they either do not share it or supply inaccurate information. Because of this issue during the data-gathering phase, the outcomes of data mining techniques are impacted, which rely on significant volumes of reliable data to provide useful findings or information. Because it facilitates the sharing of private data for analysis purposes, privacy-preserving data mining (PPDM) has grown in popularity [3].

Code profiling is a type of dynamic programmed analysis that analyses things like a program's space (memory) or time complexity, the use of certain instructions, or the frequency and length of function calls. Software code profiling data may tell us about its different features relating to its nature and performance, which is highly significant. It is a type of dynamic programme analysis in which the space (memory) or time complexity of a programme, the use of certain instructions, or the frequency and length of function calls are measured. There are several software metrics that may be found in code profiling data. Program profiling, often known as software profiling, is a type of dynamic programmed analysis (as opposed to static code analysis) that examines a program's behavior using information acquired while the programmed operates The main goal of this study is to discover which parts of a programmed should be optimized in order to boost overall performance, decrease memory requirements, or both.

Code profiling data on a project or code is highly useful to a company in this task. If it wishes to mine this data without jeopardizing its privacy, privacy-preserving mining is the ideal approach. As code profiling data, we used a variety of software measures. Using this, we establish a model by learning the software metrics of current successful projects, and then we can forecast the value of an unknown measure based on the developed model for a new project. This can benefit an organization in two ways. First, the

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

privacy of the code profiling data is maintained. Second, we may use a model based on successful projects to forecast unknown attributes for a new project, and we can use that model to identify the flaws of the new project.

LITERATURE REVIEW

A survey on privacy-preserving data mining has been proposed by Jian Wang, Yongcheng Luo, Yan Zhao, and Jiajin Le [3]. They clearly outlined numerous privacy-preserving data mining tools before analyzing their benefits and drawbacks. They've spoken about K-anonymity, the perturbation method, and cryptographic algorithms. They proposed using Randomized Response approaches to tackle the DTPD problem and provided a condensation methodology that generates pseudo-data from the statistics of these confined clusters in the data set. Pan Yang, Xiaolin Gui, Feng Tian, Jing Yao, and Jiancai Lin [2] suggested a data obfuscation strategy that protects privacy in data statistics and data mining. Yuriy Brun and Michael D. Ernst [6] proposed a method for detecting hidden code faults in programme runs using machine learning. Their method presents a method for detecting software attributes that suggest problems. This approach creates machine learning models of programme characteristics known to come from mistakes and applies these models to properties of programmes written in code to categorize and rank properties that may contribute to errors. Quinn Taylor and Christophe Giraud-Carrier [9] presented data mining applications in software engineering. Because software engineering processes are extremely complex and associated activities frequently generate a great number and diversity of artefacts, they are well-suited to data mining. In recent years, there has been a surge in the application of data mining techniques on such arte facts with the goal of Software process analysis and optimization for a given organization or project Software metrics and software complexity metrics have been studied by Tu Honglei, Sun Wei, and Zhang Yanan [4]. Their study discusses McCabe techniques and the C&K metric approach as examples of complexity metrics. They also discussed software metrics, including the concept of metrics and the history of this discipline, before moving on to complexity metrics, such as McCabe complexity metrics and object-oriented metrics, with real-world examples. Chidamber and Kemerer proposed the C&K technique in 1994. Yuriy Brun and Michael D. Ernst [5] have developed a method for detecting hidden code faults in programme runs using

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

machine learning. Their method presents a method for detecting software attributes that suggest problems. This approach creates machine learning models of programme characteristics known to come from mistakes and applies these models to properties of programmes written in code to categorize and rank properties that may contribute to errors.

PROPOSED WORK

Proposed Algorithm

Input: Data Set, key1, key2.

Step 1. Create an Encrypted Data Set with Encrypt (Data Set, Key1, Key2)

Step 2. Apply k-means clustering and k-NN classification to the Data Set and Encrypted Data Set, respectively.

Step 3. Check the accuracy of the above two outcomes.

Step 4. Analyze the Data Set's mining findings and compute the Correlation Coefficient (Data Set) between all of the characteristics.

Step 5. Remove characteristics if the correlation coefficient is low.

Step 6. To improve results, repeat k-means clustering and k-NN classification.

Output: Correlation Coefficient, Data Set Class

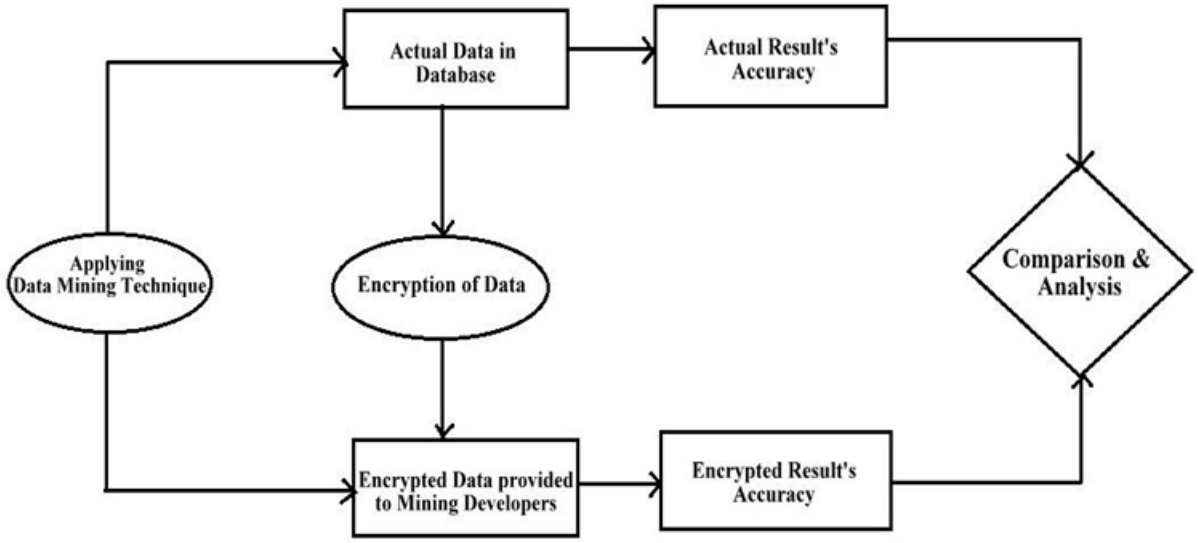


Fig 1. Stepwise block diagram of the proposed method

EXPERIMENT AND RESULTS

The suggested technique is validated using data from code profiling for 25 software metrics and code coverage for 140 Java scripts. Because it is difficult to mine source codes directly in this study, we

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

mined them indirectly using their code profiling data. Some of the Java code was created by us, while others came from free source libraries. We developed 26 code profiling attributes for each java code. Therefore the database contains 140 instances, each for code profiling data of a certain code. This data was obtained using code profiling tools Eclipse Metrics and Code Pro Analytix. The results of code profiling tools are provided below for a single java code.

Metric	Total	Mean	Std. ...	Maxi...	Resource causing Maximum	Method
Number of Overridden Methods (avg	0	0	0	0	/Cohesion/src/co/Cohes.java	
Number of Attributes (avg/max per t	2	1	1	2	/Cohesion/src/co/fun.java	
Number of Children (avg/max per typ	0	0	0	0	/Cohesion/src/co/Cohes.java	
Number of Classes (avg/max per pac	2	1	1	2	/Cohesion/src/co	
Method Lines of Code (avg/max per	8	1	0	1	/Cohesion/src/co/Cohes.java	msg1
Number of Methods (avg/max per ty	8	4	2	6	/Cohesion/src/co/fun.java	
Nested Block Depth (avg/max per m		1	0	1	/Cohesion/src/co/Cohes.java	msg1
Depth of Inheritance Tree (avg/max p		1	0	1	/Cohesion/src/co/Cohes.java	
Number of Packages	2					
Afferent Coupling (avg/max per pack		0.5	0.5	1	/Cohesion/src/coh	
Number of Interfaces (avg/max per p	1	0.5	0.5	1	/Cohesion/src/coh	
McCabe Cyclomatic Complexity (avg/		1	0	1	/Cohesion/src/co/Cohes.java	msg1
Total Lines of Code	45					
Instability (avg/max per packageFrag		0.5	0.5	1	/Cohesion/src/co	
Number of Parameters (avg/max per		0	0	0	/Cohesion/src/co/Cohes.java	msg1
Lack of Cohesion of Methods (avg/m		0.25	0.25	0.5	/Cohesion/src/co/fun.java	
Efferent Coupling (avg/max per pack		0.5	0.5	1	/Cohesion/src/co	
Number of Static Methods (avg/max	0	0	0	0	/Cohesion/src/co/Cohes.java	
Normalized Distance (avg/max per p		0	0	0	/Cohesion/src/co	

Fig 2. Software metrics values generated by Eclipse Metrics tool

CodePro Code Coverage for decrypt.Check

This document contains the code coverage results for decrypt.Check.

Code Coverage Summary — **66.7% coverage**

Line Coverage	66.7%
Block Coverage	60.0%
Instruction Coverage	82.6%

Code Statistics — **1 classes, 6 executable lines**

Total Packages	1
Total Files	1
Total Classes	1
Total Executable Lines	6

Fig 3. Code Coverage generated by Code Pro Analytix tool

As we can see, this output is in GUI format and cannot be mined directly. Therefore we had to turn it into tangible data form, so we extracted the data's XML file. We turned this GUI data into actual data form and saved it in MS Excel sheets

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

using JAXP (Java API for XML Processing). On this data, privacy-preserving data mining algorithms are used. These two tools are used to create software metrics for Java code. The output of these tools is in the GUI (Graphical User Interface) format. Then we retrieved the XML file of this GUI format and used JAXP to extract the software metrics values from the XML file. These values are recorded in a database for the codes, and mining is then performed on this database.

To protect the privacy of code profiling data, we encrypted it with a basic encryption method and two keys. Then, using both encrypted and unencrypted data, k-means clustering is used to generate 5 groupings. A comparison is made between both data sets to ensure that anonymity is maintained while mining results remain consistent.

We know the class of each instance once we generate these clusters. We compared the real and encrypted classification results using 10-fold cross-validation and a k-NN classifier. Encryption protects privacy, and mining results are also extremely accurate and correct. There is no difference between the real and encrypted data outcomes. We compared these clusters to the features of our data and discovered that they are mostly dependent on two attributes: Code Coverage and LOC. We discovered that certain clusters are located in unexpected clusters. For example, we created inheritance-related programmers, and all but one of them are in the same cluster. Because of this unanticipated data behavior, we estimated the correlation coefficient between all of our dataset's properties. We determined correlation coefficients for several critical features in all cases, and then deleted those attributes that were not positively associated, and clustering was conducted again for improved results with greater accuracy and precision. We discovered that by deleting weakly correlated data, we enhanced the clustering approach while decreasing the accuracy of the classifier.

As a consequence, we may conclude that by deleting weakly correlated characteristics from code profiling data, we can increase clustering while decreasing classifier accuracy.

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

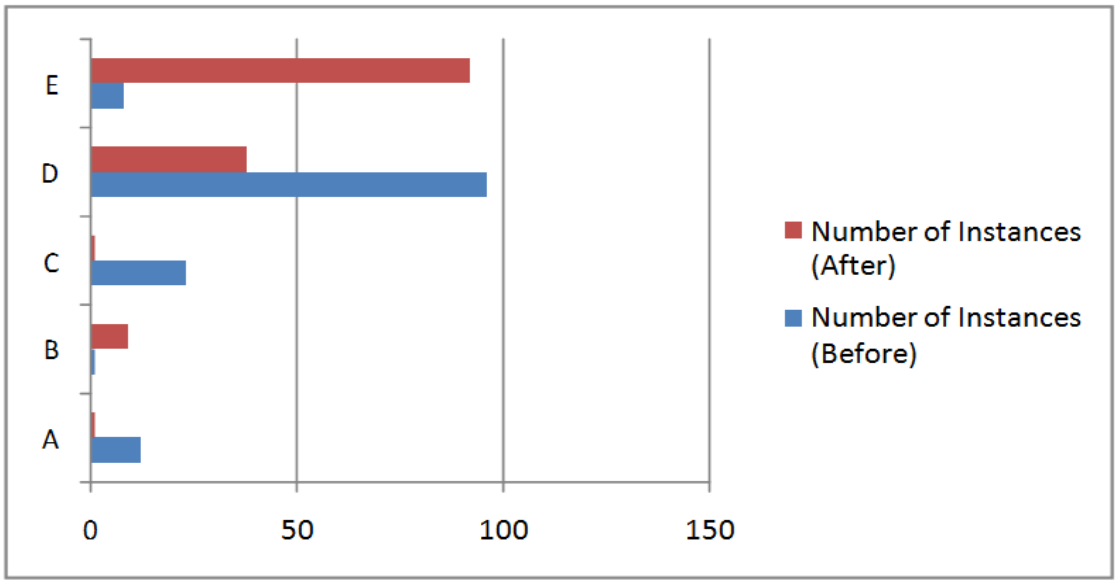


Fig. 4. Comparison graph between clusters before vs. after

CONCLUSION AND FUTURE SCOPE

We used code profiling tools such as Code Pro Analytix and Eclipse Metrics to generate code profiling data for 140 java scripts. Following that, we translated the GUI data into the real data from using JAXP. That is how we obtain our pre-processed database, on which we have performed privacy-preserving mining. To protect the privacy of this data, we used an encryption technique on it and built an encrypted version of it. Then we used k-means clustering and 10-fold cross-validation to determine the accuracy of the k-NN classifier on this data using Euclidean distance and examined the findings.

Mining for privacy is currently a developing study subject in the realm of software engineering. Various companies have massive amounts of sensitive data regarding their projects that may be mined to gain knowledge for future advancements in this industry. The software creates massive amounts of data that may be mined for insight. As a result, various combinations of encryption and classification approaches may be utilized to increase the accuracy of privacy-preserving mining in code profiling data.

REFERENCES

1. R. Mitchell & R. Chen. (2014). Adaptive intrusion detection of malicious unmanned air vehicles using behaviour rule specifications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 44(5), 593–604.

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved

2. Jian Wang, Yongcheng Luo, Yan Zhao, & Jiajin Le. (2009). A survey on privacy-preserving data mining. In 1st International Workshop on Database Technology and Applications, 111-114.

3. Quinn Taylor & Christophe Giraud-Carrier. (2010). Applications of data mining in software engineering. International Journal of Data Analysis Techniques and Strategies, 2(3), 243-257.

4. Antal, P., Fannes, G., Timmerman, D., Moreau, Y., & De Moor, B. (2003). Bayesian applications of belief networks and multilayer perceptrons for ovarian tumour classification with rejection. Artificial Intelligence in Medicine, 29, 39-60.

5. Tu Honglei, Sun Wei, & Zhang Yanan. (2009). the research on software metrics and software complexity metrics. In International Forum on Computer Science- Technology and Applications, 131-136.

6. Yuriy Brun & Michael D. Ernst. (2004). Finding latent code errors via machine learning over program executions. In 26th International Conference on Software Engineering (ICSE), 480-490.

Page 23-30 © MANTECH PUBLICATIONS 2022. All Rights Reserved