www.ijemr.net ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Vandana Publications
IJEMR

Volume-8, Issue-5, October 2018
International Journal of Engineering and Management Research
Page Number: 24-28
DOI: doi.org/10.31033/ijemr.8.5.5

Privacy Preserving Mining in Code Profiling Data

Meenakshi Kathayat
Assistant Professor, Department of Computer Science & Engineering, Birla Institute of Applied Sciences Bhimtal, Uttarakhand, INDIA

Corresponding Author: meenakshik.kathayat4@gmail.com

ABSTRACT

Privacy preserving data mining is an important issue nowadays for data mining. Since various organizations and people are generating sensitive data or information these days. They don't want to share their sensitive data however that data can be useful for data mining purpose. So, due to privacy preserving mining that data can be mined usefully without harming the privacy of that data. Privacy can be preserved by applying encryption on database which is to be mined because now the data is secure due to encryption. Code profiling is a field in software engineering where we can apply data mining to discover some knowledge so that it will be useful in future development of software. In this work we have applied privacy preserving mining in code profiling data such as software metrics of various codes. Results of data mining on actual and encrypted data are compared for accuracy. We have also analyzed the results of privacy preserving mining in code profiling data and found interesting results.

Keywords-- Privacy preserving data mining, Code Profiling, Correlation coefficient

I. INTRODUCTION

Software systems are mainly complex and hard to conceptualize. This complexity, compounded by complicated dependencies and different programming practices, slows development and maintenance activities, leads to faults and defects and finally increases the cost of software [1]. Software engineering activities generate a huge amount of data that, if harnessed properly through data mining techniques, can help provide awareness into many parts of software development processes. Privacy preserving data mining is important because of the huge amount of personal data generated nowadays. Privacy preserving is an important issue in the data mining field. Many applications are benefited from data sharing, mainly data statistics and data mining. But the shared data may contain private information of the owner of data. It has a high risk of revealing data owner's privacy [2]. Most of the times people are not interested in sharing their private data, they either not share their data or provide incorrect data. Due to this problem in data collection phase, the results of data mining techniques gets affected, which is based on sufficient amounts of accurate data for producing meaningful results or knowledge. Privacy preserving data mining (PPDM) has become increasingly popular because it allows sharing of private data for analysis purposes [3].

Code profiling is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the use of particular instructions, or the frequency and duration of function calls. Code profiling data of software can tell us about its various attributes regarding its nature and performance and is very important. It is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the use of particular instructions, or the frequency and duration of function calls. There are various software metrics which can come in code profiling data. In software engineering, program profiling or software profiling is a form of dynamic program analysis (as opposed to static code analysis) or the examination of a program's behavior by gathered information as the program executes. The usual purpose of this analysis is to determine the sections of a program needed to optimize - to increase its overall speed, decrease its memory requirement or sometimes both.

In this work Code profiling data of a project or code is very important to an organization. If it wants to mine this data without risking its privacy then privacy preserving mining is the best option for that purpose. We have taken various software metrics as code profiling data. Using this we create a model by learning existing successful project's software metrics and then for a new project we can predict the value of an unknown metrics

based on the generated model. This can help an organization in 2 aspects-First, the privacy of the code profiling data is preserved. Second, we can predict unknown attribute for a new project using a model based on successful projects and we can know the shortcomings of the new project on that basis.

II. LITERATURE REVIEW

Jian Wang, Yongcheng Luo, Yan Zhao, and Jiajin Le [3] have proposed a survey on privacy preserving data mining. They have described several privacy preserving data mining technologies clearly and then analyzed the merits and shortcomings of these technologies. They have discussed K-anonymity, the perturbation approach, Cryptographic techniques. They have suggested to use the Randomized Response techniques to solve the DTPD problem and introduced a condensation approach, which creates constrained clusters in the data set, and then generates pseudo-data from the statistics of these clusters. Pan Yang, Xiaolin Gui, Feng Tian, Jing Yao, and Jiancai Lin [2] have proposed a privacy-preserving data obfuscation scheme used in data statistics and data mining. Yuriy Brun, and Michael D. Ernst [6] have introduced a technique for finding latent code errors via machine learning over program executions. Their technique proposes a technique for identifying program properties that indicate errors. This technique generates machine learning models of program properties known to result from errors, and applies these models to properties of programs of code written to classify and rank properties that may lead the user to errors. Quinn Taylor, and Christophe Giraud-Carrier [9] have introduced applications of data mining in software engineering. Since software engineering activities are very complex, and the related activities often produce a large number and variety of artefacts, so that they are well-suited to data mining. Recent years have seen an increase in the use of data mining techniques on such artefacts with the aim of analysing as well as improving software processes for a given organisation or project.

Tu Honglei, Sun Wei, and Zhang Yanan [4] have done a research on software metrics and software complexity metrics. Their work respectively expounds McCabe methods and C&K metric method for examples of complexity metrics. They also introduced the software metrics inclusive of the definition of metrics and the history of this field; then brought up the complexity metrics, such as McCabe complexity metrics and object oriented metrics, with real world examples. C&K method was brought up in 1994 by Chidamber and Kemerer. Yuriy Brun, and Michael D. Ernst [5] have introduced a technique for finding latent code errors via machine learning over program executions. Their technique proposes a technique for identifying program properties that indicate errors. This technique generates machine learning models of program properties known to result from errors, and applies these models to properties of programs of code written to classify and rank properties that may lead the user to errors.

III. PROPOSED WORK

3.1 Proposed Algorithm

Input: Data Set, key1, key2.
Step 1. Generate Encrypted_Data Set using Encrypt (Data Set, Key1, Key2).
Step 2. Apply k-means clustering (Data Set, Encrypted_Data Set) and k-NN classification (Data Set, Encrypted_Data Set).
Step 3. Compare the results of above two for correctness.
Step 4. Analyze mining results of the Data Set and calculate Correlation_Coefficient (Data Set) between all the attributes.
Step 5. If correlation coefficient weak → Remove attributes.
Step 5. Apply k-means clustering and k-NN classification again to get better results.
Output: Data Set_Class, Correlation_Coefficient.

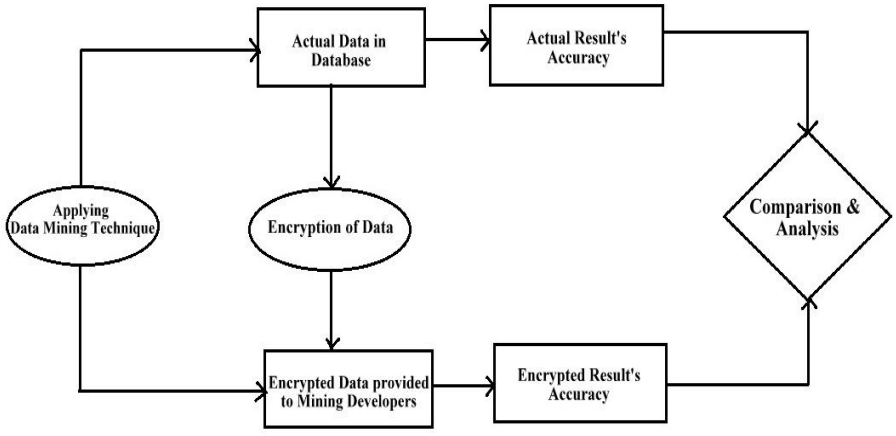

Fig 3.1 Step wise block diagram of the proposed method.

ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

IV. EXPERIMENT AND RESULTS

The proposed method is tested by code profiling data of 25 software metrics and code coverage for 140 java codes. Source codes are not mined directly in this work because it is very complex to do that, that's why we indirectly mined them using their code profiling data. Some of the java codes are written by us and some are taken from open source libraries. For each java code we have generated 26 code profiling attributes, hence the database consists of 140 instances, each for code profiling data of a particular code. We have generated this data by code profiling tools namely Eclipse Metrics and CodePro Analytix. The outputs generated by code profiling tools are as shown 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 me		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 4.1 Software metrics values generated by Eclipse Metrics tool

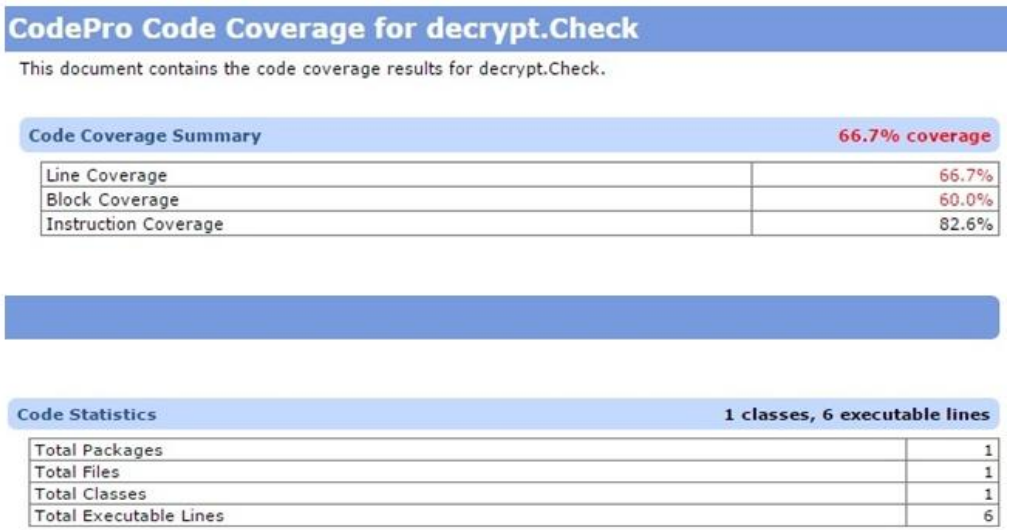

Fig 4.2 Code Coverage generated by CodePro Analytix tool

ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

As we can see this output is in GUI form and cannot be directly mined. So we had to convert it into concrete data form, so we extracted the XML file of this data. Through JAXP (Java API for XML Processing) we have converted this GUI data into actual data form and saved it into MS Excel sheets. The privacy preserving data mining techniques are applied on this data. Software metrics are generated for the codes written in java using these two tools. These tools generate output in GUI (Graphical User Interface) format. Then we have extracted XML file of this GUI format and through JAXP we have extracted the software metrics values from XML file. These values are saved as database for the codes and then mining is applied on this database.

For preserving the privacy of code profiling data, we have encrypted this data using simple encryption algorithm with 2 keys. Then k-means clustering is applied to create 5 clusters, on encrypted data as well as on actual data. Comparison is made between both data for confirming that privacy is preserved and at the same time mining results do not vary.

After generating these clusters we know the class of each instance. Through 10-fold cross validation and k-NN classifier we have compared the actual and encrypted results of classification technique. Privacy is preserved by encryption and mining results are also very good and correct. There is not any variation between actual and encrypted data results. We analyzed these clusters with the properties of our data and found that the clusters are majorly based on two attributes: Code Coverage and LOC. We found that some clusters lie in unexpected clusters. For example, we made inheritance related codes and all of them are in same clusters except one of them. Due to this unexpected behavior of data we calculated correlation coefficient between all the attributes of our dataset. For all instances we calculated correlation coefficient for various important attributes, then we removed all those attributes which are not correlated positively and again clustering is performed for better results with more accuracy and precision. On analyzing we found that on removing weakly correlated data, we improved the clustering method but at the same time classifier's accuracy is decreased.

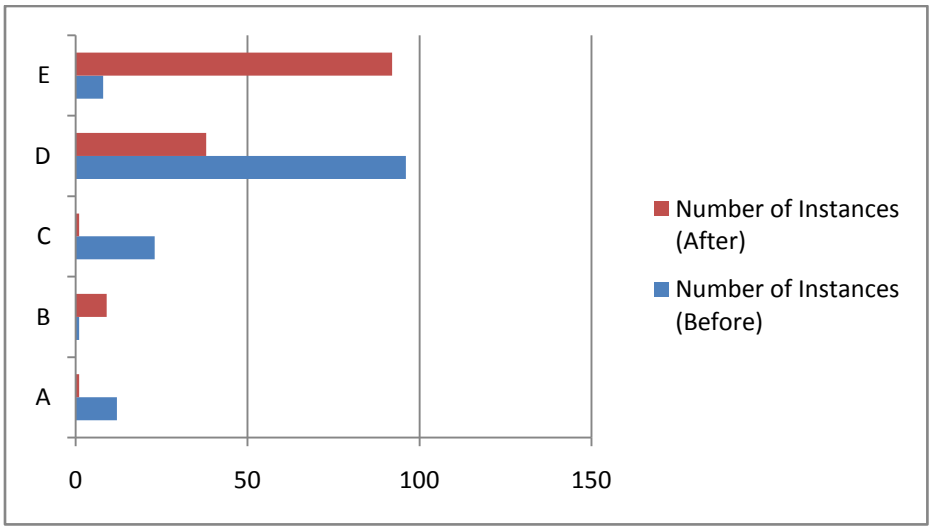

Fig 4.3 Comparison graph between clusters before vs. after

So, it can be concluded that by removing weakly correlated attributes of code profiling data we can improve the clustering but at the same time they are resulting into decrease in accuracy of classifier.

V. CONCLUSION AND FUTURE SCOPE

We have pre processed 140 java codes to generate their code profiling data using code profiling tools such as, CodePro Analytix and Eclipse Metrics. After that we have converted this GUI data through JAXP in the actual data form. That's how we get our pre processed database on which we have applied privacy preserving mining. For preserving privacy of this data we have applied encryption technique on this data and generated its encrypted form. Then we have applied k-means clustering and 10-fold cross validation to find accuracy of k-NN classifier on this data through Euclidean distance and analyzed the results thereafter.

Privacy preserving mining is still an emerging research area in the field of software engineering. Various organizations have enormous amount of sensitive data about their projects which can be mined to get knowledge for future improvement in this field. Software generates huge amount of data which can be mined to discover knowledge. So, different combinations of encryption

methods and classification methods can be used to improve the accuracy of privacy preserving mining in code profiling data.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] A.V.Krishna Prasad & Dr. S.Rama Krishna. (2010). Data mining for secure software engineering – Source code management tool case study. *International Journal of Engineering Science and Technology*, *2*(7), 2667-2677.