IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 8, Issue.12, December 2020 E-ISSN: 2347-2693

Improve the accuracy and time complexity of code smell detection using SVM and DECISION-TREE with MULTI-LABEL CLASSIFACTION

Manpreet Kaur[1*], Deepinder Kaur[2]

DOI: https://doi.org/10.26438/ijcse/v8i12.6669 | Available online at: www.ijcseonline.org

Received: 04/Dec/2020, Accepted: 11/Dec/2020, Published: 31/Dec/2020

Abstract: Code smell refers to an anomaly in the source code that shows violation of basic design principles such as abstraction, hierarchy, encapsulation, modularity. In this research we are using SVM (support vector Machine) and decision Tree for code smell detection. In this research we improving the accuracy and time complexity of error in code with the help of Multi-Label classification.

Keywords: CODE SMELLS, VECTOR MACHINE

I. INTRODUCTION

Code smell refers to an anomaly in the source code that shows violation of basic design principles such as abstraction, hierarchy, encapsulation, modularity etc. in other words we can say Code smells refer to any symptom in the source code of a program that possibly indicates a deeper problem, hindering software maintenance and evolution. Detection of code smells is challenging for developers and their informal definition leads to the implementation of multiple detection techniques and tools. Developers are typically trained to look out for logical errors that have been accidentally introduced to their code. Such errors will range from forgotten edge cases that have not been handle to logical bugs that cause entire systems to crash. Code Smells are signals that your code should be re-factored in order to improve extendibility, readability, and supportability.

1.1 TYPES OF CODE SMELLS:

Some common code smells are found in project codes.

1. Bloaters

Bloaters are code, methods and classes that have increased to such proportions that they are hard to work with. Usually these smells do not crop up right away, rather they accumulate over time as the program evolves. For example: Long Method, Large Class, Primitive Obsession, Long Parameter List, Data Clumps.

(i) **Long Method:** The majority of a programmer's time is spent reading code rather than writing code. Apart from the difficulty of having to keep a lot of complex logic in mind whilst reading a long method, it is usually a sign that the method has too many responsibilities. Long methods make code hard to maintain and debug. If it is not possible to view the whole method on your smart phone screen, consider breaking it up into several smaller methods, each doing one precise thing.

(ii) **DataClumps:** Where multiple method calls take the same set of parameters, it may be a sign that those parameters are related. To keep the group of parameters together, it can be useful to combine them together in a class. This can help aid organization of code.

2. Object Oriented Abusers:

All these smells are incomplete or incorrect application of object-oriented programming principles. For example, Switch Statements, Temporary Field, Refused Bequest, Alternative Classes with Different Interfaces.

3. Change-Preventers

These smells mean that if you need to change something in one place in your code, you have to make many changes in other places too. Program development becomes much more complicated and expensive as a result. For example: Divergent Change, Shotgun Surgery, Parallel Inheritance Hierarchies:

(i) Duplicate-Code:

When developer fixes a bug, but same symptoms are faced again later on, this can be the result of code duplication, and a bug being fixed in one occurrence of the imperfect code but not in the duplicated versions. This poses an overhead in terms of maintenance. When developers are not aware of the duplication, they only know to fix the occurrence they have come across. Take care of the repeated code blocks and extract them out into a single place – don't repeat yourself.

(ii) Inheritance-method:

If a class inherits from a base class but doesn't use any of the inherited fields or methods, developers should ask themselves if inheritance really is the right model. Signs of this code smell may be that the inherited methods go unused, or are over ridden with empty method parts.

Inheritance should be used when a class wants to reuse the code in its super class. If the classes diverge and the subclass no longer needs that functionality, the hierarchy should be broken and delegation considered instead. And to keep some inheritance, remove the unused fields and methods from the subclass and create a new layer that the objects can inherit from.

II. MACHINE LEARNING

2.1 SUPPORT VECTOR MACHINE:

A support vector machine (SVM) is machine-learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs used in text categorization, image classification, handwriting recognition and in the sciences.

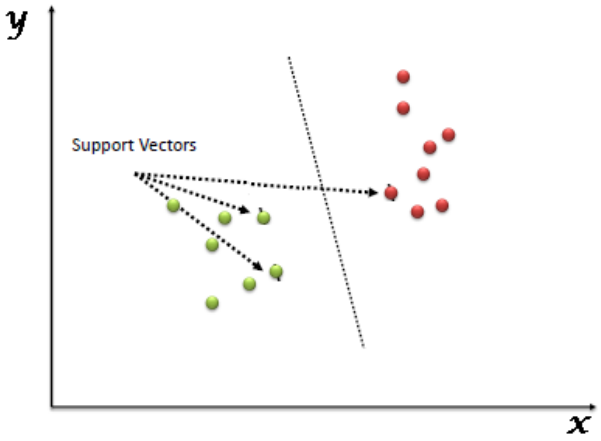


Figure 1.3 SVM A support vector machine is known as a support vector network (SVN).

2.2 DECISION TREE:

Decision Tree learning algorithm generates decision trees from the training data to solve classification and regression problem. Decision Tree Algorithm uses some steps:

1. Start with a training data set which we'll call S. It should have attributes and classification.
2. Determine the **best attribute** in the dataset. (We will go over the definition of best attribute)
3. Split S into subset that contains the possible values for the best attribute.
4. Make decision tree node that contains the best attribute.
5. Recursively generate new decision trees by using the subset of data created from step 3 until a stage reached where you cannot classify the data further. Represent the class as leaf node.

III. REVIEW OF LITERATURE

There are we will discuss the research of some researchers that worked on code smell and try to detect the code smell detection.

1. **Guggulothu et.al [2019]** researched on code smell and give a new machine learning algorithm which work on multi-label classification for detection of code smell. This multi- label technique is use to remove the errors from multiple lines at one time. This technique is enhanced technique for single line detection algorithm. These researchers only focused only remove the errors in less time or we can say that this researcher only work on time complexity. This is the major drawback of this research that this technique not give the accurate result of detection of code sometimes the inherited code do not detect by this technique.[1]

2. **Xinghua et.al [2016]** researched on code smell detection tools, these researchers gives a new tool for detection of code smell. They researched a new tool name called DT which work on automatically detection of code smell from a code when errors comes. This algorithm is better than iplasma,PMD,checkstyle named tools which were used for detection of errors in a code. The major disadvantage of this tool that it only worked on 11 types of errors not on the every types of errors this is the major drawback of this research.[2]

3. Muhammad et.al [2019] worked on machine learning to detect the code smell. These researcher only detect the long method code detection by using SVM(support vector machine) technique. This is one of useful technique for smell detection in a code, but this researcher only use it on a particular type of error. This is not useful for the all the other type of errors.[3]

4. Paiva et.al [2017] researched on code smell detection by using four different detection tools. These four detection tools were nFusion, JDeodorant, PMD, and JSpIRIT. These technique used on different versions of same software. These four tools detect the three different types of code smells like : God class, God Method and feature envy. All these techniques works on these techniques but only in the form classes, these tools only work on classes rather than objects.[4]

5. Mariani et.al[2011] researched on different techniques of different projects, these researchers gave a brief survey of different smell detection techniques for code detection. These techniques worked on accuracy of the data. Not all these technique presented a useful tool or technique for smell code detection. Only all the techniques worked on GUI of the code.[5]

IV. PROBLEM FORMULATION

In previous research, researchers only worked on code smell and give a new machine-learning algorithm which work on multi-label classification for detection of code smell. This multi- label technique is use to remove the errors from multiple lines at one time. This technique is enhanced technique for single line detection algorithm. These researchers only focused only remove the errors in less time or we can say that this researcher only work on time complexity. This is the major drawback of this research that this technique not give the accurate result of detection of code sometimes the inherited code do not detect by this technique. In proposed technique we try to enhance the technique for code smell detection by using combination of multi-label classification and decision tree and SVM.

OBJECTIVES:

1. Detect the code smell detection
2. Improve the accuracy of the code using decision tree
3. Improve the time complexity by using multilabel detection
4. Detect the smell or error from the code by using SVM technique of Machine Learning.
5. Compare the accuracy of previous research which was done only using multi label detection and our new Enhanced algorithm by using SVM.

V. RESEARCH METHODLOGY

In proposed technique, we will try to improve the drawbacks of existing technique by using following steps.

1. Select the dataset.
2. Start finding the smell from code by using Decision tree classification.
3. Make a Tree of smells from the code.
4. Use the multi-label classification for detect the smell from the code.
5. Make the detected code.
6. Apply the SVM (support vector machine) for final classification on the code give accurate result.
7. Propose a new algorithm by using these techniques.

VI. RESULTS AND DISCUSSION

A support vector machine (SVM) is machine-learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories.

Find the Training data set:

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

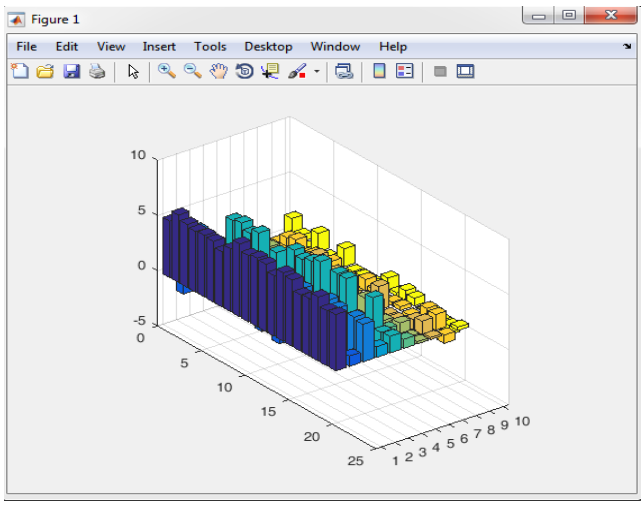
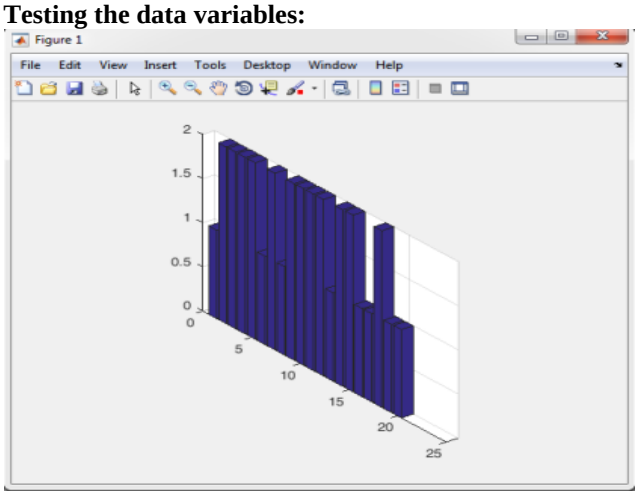
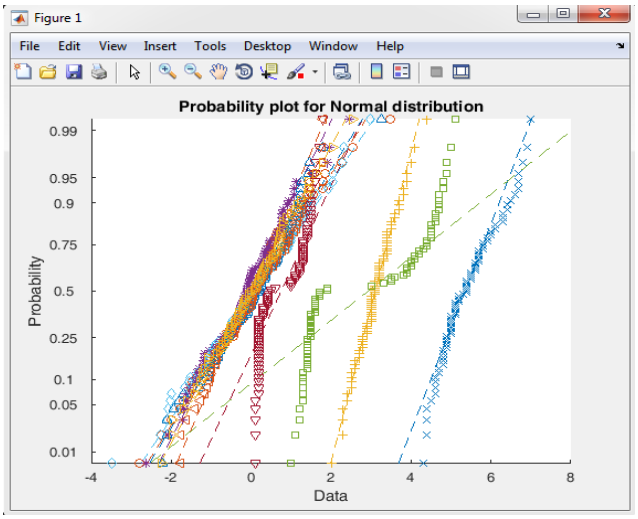


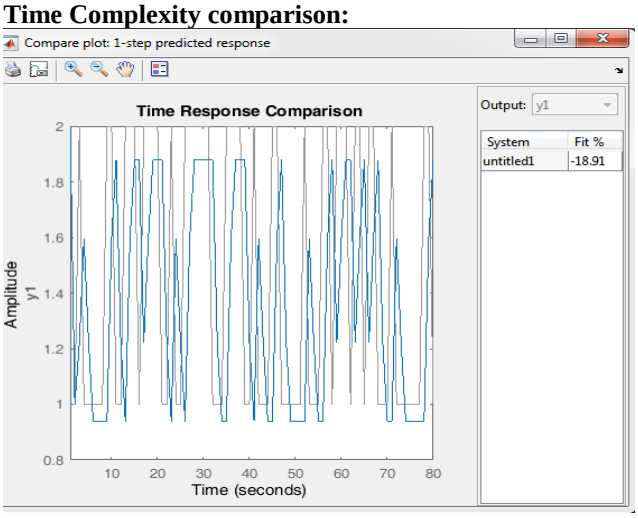
Figure 2

Testing the data variables:





Time Complexity comparison:



VII. CONCLUSION

In this Research, we try to improve the accuracy of code smell detection. Smell in a code means errors in code .In this research we improve the time complexity as well which take less time to find out an error from the large code.

REFERENCES

[1] Thirupathi Guggulothu, Salman Abdul Moiz_Code Smell Detection using Multilabel Classi_cation Approach,School of Computer and Information Sciences, University of Hyderabad, Hyderabad-500 046, Telangana, India

[2] DT : a detection tool to automatically detect code smell in software project Xinghua Liu1, a and Cheng Zhang2, b 1 School of Computer Science and Technology, Anhui University, China 2 School of Computer Science and Technology , Anhui University, China a xinghua.liu@ahu.edu.cn , b cheng.zhang@ahu.edu.cn

[3] Information and Software Technology,Volume 108, April 2019, Pages 115-138 "Machine learning techniques for code smell detection: A systematic literature review and meta-analysis" Muhammad IlyasAzeem[ab]FabioPalomba[d]LinShi[ab]QingWang[abc],Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

[4] "On the evaluation of code smells and detection tools" ,Thanis Paiva, Amanda Damasceno, Eduardo Figueiredo & Cláudio Sant'Anna ,Journal of Software Engineering Research and Development volume 5, Article number: 7 (2017)

[5]An experience report on using code smells detection tools Francesca Ar[5]Università of Milano Bicocca Department of Computer Science Milano, Italy arcelli@disco.unimib.it ,Andrea Morniroli, Raul Sormani, Alberto Tonello ,University of Milano Bicocca Department of ComputerScience Milano, Italy a.morniroli@campus.unimib.it

[6]https://becominghuman.ai/decision-trees-in-machine-learning-f362b296594a

