Development of Models for Improving Software Maintenance

A Dissertation submitted in the partial fulfillment for the award of

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

by

Ashish Kumar Tripathi

Roll no. 2k11/SWE/04

Under the Essential Guidance of

Dr. Ruchika Malhotra

Department of Computer Engineering

Delhi Technological University

New Delhi

2012-2013

DECLARATION

I hereby declare that the thesis entitled "**Development of Models for Improving Software Maintenance"** which is being submitted to the **Delhi Technological University**, in partial fulfillment of the requirements for the award of degree of **Master of Technology in Software Engineering** is an authentic work carried out by me. The material contained in this thesis has not been submitted to any university or institution for the award of any degree.

_____

Ashish kumar tripathi

Department of Computer Engineering

Delhi Technological University,

Delhi.

CERTIFICATE

DELHI TECHNOLOGICAL UNIVERSITY

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI-110042

Date: _____

This is to certify that the thesis entitled "**Development of Models for Improving Software Maintenance "**submitted by **Ashish kumar tripathi(Roll Number: 2K11/SWE/04),** in partial fulfillment of the requirements for the award of degree of Master of Technology in Software Engineering, is an authentic work carried out by him under my guidance. The content embodied in this thesis has not been submitted by him earlier to any institution or organization for any degree or diploma to the best of my knowledge and belief.

Project Guide

Dr. Ruchika Malhotra

Assistant Professor

Department of Computer Engineering

Delhi Technological University, Delhi-110042

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this thesis.

Foremost, I would like to express my sincere gratitude to my guide **Dr. Ruchika Malhotra**,**Assistant Professor,Department of Computer Engineering**, **Delhi Technological University, Delhi** whose benevolent guidance, constant support, encouragement and valuable suggestions throughout thecourse of my work helped me successfully complete this thesis. Without her continuous support and interest, this thesis would not have been the same as presented here.

Besides my guide, I would like to thank the entire teaching and non-teaching staff in the Department of Computer Science, DTU for all their help during my course of work.

ASHISH KUMAR TRIPATHI

Table of Contents

CHAPTER 1 1

INTRODUCTION 1

1.1. SOFTWARE MAINTENANCE 2

1.2. SOFTWARE QUALITY 2

1.3. SOFTWARE METRICS 2

1.4. MOTIVATION 3

1.5. ORGANIZATION OF THESIS 5

CHAPTER 2 6

LITRATURE SURVEY 6

CHAPTER 3 9

RESEARCH BACKGROUND 9

3.1. CHANGE PRONENESS 9

3.2. METRICS SELECTED FOR STUDY 9

3.3. INDEPENDENT AND DEPENDENT VARIABLES 10

CHAPTER 4 18

RESEARCH METHODOLOGY 18

4.1. EMPIRAICAL DATA COLLECTION 18

4.1.1 STATICS FOR J-METER 18

4.1.2 STATICS FOR XML SECURITY 22

4.2. CALCULATION OF CHANGES IN THE TWO DIFFERENT VERSIONS 24

4.3. DATA ANALYSIS FOR CHANGE PRONE CLASSES 26

4.4. METHODS USED 28

CHAPTER 5 35

RESULTS ANALYSIS 36

5.1. UNIVARIATE LR ANALYSIS RESULTS 36

5.2. MULTIVARIATE LOGISTIC REGRESSION ANALYSIS OF J-METER 38

5.3. MULTIVARIATE LOGISTIC REGRESSION ANALYSIS OF XML-SECURITY 39

5.4. MODEL EVALUATION USING THE ROC CURVE 40
5.5. INTER PROJECT VALIDATION OF PREDICTION USING J-SECURITY 55
5.6. DISCUSSION OF RESULTS 56

CHAPTER 6 57
CONCLUSION AND FUTURE WORK 57

REFERENCES 59

List of Figures

Figure 4.1 Change analysis of J-Meter.....20

Figure 4.2 Change analysis of XML-security 23

Figure 4.3 Updation rule 25

Figure 4.4 Data Analysis Life cycle.....27

Figure 5.1 Bayes Net curve 42

Figure 5.2 Naïve Bayes curve.....43

Figure 5.3 J48 curve 43

Figure 5.4 Random forest curve 44

Figure 5.5 Adaboost curve 44

Figure 5.6 Bagging curve 45

Figure 5.7 Part curve.....45

Figure 5.8 Conjuctive rule curve 46

Figure 5.9 K star curve.....46

Figure 5.1. nnge curve.....47

Figure 5.11 Logistical regression curve. 47

Figure 5.12 Forward LR curve. 49

Figure 5.13 Bayes Net curve. 50

Figure 5.14 Naïve Bayes curve.....50

Figure 5.15 J48 curve.....51

Figure 5.16 Random forest curve.....51

Figure 5.17 Adaboost curve.....52

Figure 5.1. Bagging curve. 52

Figure 5.19 Part curve. 53

Figure 5.20 Conjuctive rule curve. 53

Figure 5.21 K star curve. 54

Figure 5.22 nnge curve. 54

List of Tables

Table 3.1 List of metrics.....11

Table 4.1 Summary of Dataset used 18

Table 4.2 Descriptions of changes for J-meter 21

Table 4.3 Descriptions statistics of J-meter.....21

Table 4.4 Statistics for XML security.....22

Table 4.5 Descriptions statistics of XML security.....23

Table 5.1Univariate Analysis of J-Meter 37

Table 5.2Univariate Analysis of XML security 37

Table 5.3 Multivariate model statics for J-Meter 39

Table 5.4 Multivariate model statics for XML-Security.....39

Table 5.5 Validation Result for j-Meter.....40

Table 5.6 Validation Result for j-security.....48

Table 5.7 Inter project validation result 55

ABSTRACT

With the emerging technology change now a days softwares are going through a lot of change resulting in development of so many versions and hence it is quite difficult to maintain the quality of the software. In this project we will develop models by applying machine learning and statistical techniques on the object oriented metrics to find out the change prone classes .We find relationship between object-oriented metrics given by Chidamber and Kemerers and change proneness. The results are validated using open source softwares. We will also compare and assess various machine learning techniques for predicting change in a class.The performance of the predicted models was evaluated using receiver operating characteristic analysis. By assessing the results we found out that machine learning techniques are better in comparison to statistical techniques .Another analysis showed that models constructed for a software can also be used to predict change proneness of classes of other software in same domain.

Chapter 1

INTRODUCTION

An understanding of quality attributes is relevant for the software organization to deliver high software reliability. An empirical assessment of metrics to predict the quality attributes is essential in order to gain insight about the quality of software in the early phases of software development and to ensure corrective actions as well as to make maintenance easier.

1.1 Software Maintenance

Software maintenance is the post delivery activity that costs 40-70% of the software and hence it is very important for us to manage it properly .It is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanism must be developed for evaluation, controlling and making modifications. The purpose is to preserve the quality of software over the time. The quality can be enhanced by expanding the customer base, meeting additional requirements, becoming easier to use, more efficient and employing newer technology

Maintenance is of 4 types.

- Adaptive : It is done when the environment of the software goes through changes
- Perfective: It is done to enhance and improve the functionality of the software.
- Corrective: Founding and correcting errors by the users.
- Preventive: This type of maintenance is done to enhance the reliability of the system and to prevent future problems.

1.2 Software Quality

In the context of software engineering, software quality measures how well software is designed and how well the software conforms to that design. Since software quality is also very much related to the internal structure of the software that's why it is important to analyze the structure before the delivery of the software. For the object oriented softwares finding the relationship among the object oriented metrics and change proneness and play a good role in maintain the quality of the software.

Software quality mainly includes three things.

- Conformance to requirements
- Fitness to the purpose
- Level of satisfaction

1.3 Software Metrics

It is a quantitative measure of degree to which a system, component or process possesses a given attribute. These metrics plays a important role in the software development and maintenance. Metrics can be classified into three categories namely process, project and product.

- Process metrics: These are metrics which are concerned with software development life cycle (SDLC). They can be used to improve the process efficiency of the SDLC.

- Project metrics: These metrics are more relevant to a project team as it measures the efficiency of a project team or any other tools being used by team members

- Product metrics: It has more meaning in prospective of the software product being developed. One of the examples is quality of the developed product.

1.4 Motivation

With the emerging demand of quality and maintainability it is quite important that we should give focus to these two factors but it is a challenging task to fulfill these criteria within the limited number of resources like cost, time and budget. Hence it is necessary to have proper planning at the early phases of development regarding the maintenance of the software because changes in the future are inevitable due to defects and the enhancements in the functionality of the product .In this paper we find relationship between the object oriented metrics and change proneness by applying machine learning techniques. There are many metrics that have been proposed and used for the prediction of change proneness at class level [9] at the same time these are also empirically validated by few researchers[10].Due to scarcity of resources, complexity of software ambiguity in the requirements it is quite difficult to make change free software but these changes cause so many problems in the maintenance and also consumes much part of our budget and sometime this problem may cost a lot and also lead to the failure of the product.

So in order to analyze some metrics we have taken two open source software J-Meter and XML-Security [8]. These softwares are developed in Java language and have 335,159 classes

respectively. Since these are Open Source softwares everyone have privilege to study and also have right to modify the source code without paying any royalties to previous developers. We have used one statistical method (logistic regression) and ten machine learning methods (j48, navebayes, random forest, adaboost, bagging, part, kstar, conjunctive rule ,nngeand bays net).

Also we have analyzed the performance by calculating area under the Receiver Operating

Characteristic (ROC) curve [6]. Receiver Operating Characteristic curve finds the balance between the number of classes predicted being change prone, and the number of classes that are not change prone.

In the past few years so many companies have started to introduce object-oriented technology for software development analysis and design. The insertion of 0bject 0riented technology in the software industry at the same time has also created new challenges for companies. For this reason the metrics which reflect the specificities of the 0bject 0riented paradigm must be studied defined and validated to get some relationship with the change proneness and the quality of the product. It is seen that the relationship among the classes of object oriented languages like inheritance and some basic principals like polymorphism may lead several problems if some modifications are being done in the source code so these object oriented metrics must be analyzed at the beginning phases so that we can overcome with many problem in the maintenance phase and also by doing this we will be able to improve the quality as well. Although there is a number of objects oriented metrics such as CK metrics [9], QMOOD metrics [4], MOOD [5], but these are not sufficient to solve the purpose thus, it

is very important to understand the relationship of object oriented metrics and change proneness.

1.5 Organization of Thesis

Chapter 2 summarizes the work done in literature. It highlights the major contributions and research work carried out in the field of change proneness prediction using object oriented metrics.

Chapter 3 provides the detailed description of the research methods. It describes the dependent and independent variables and then focuses on to the process used to capture the details of the metrics used.

Chapter 4 explains the work carried out in this in detail with each and every step shown conceptually and diagrammatically. It emphasizes on the empirical work carried out using various industrial data sets.

Chapter 5 compares and analyzes the performance of various machine learning and statistical techniques in predicting change prone classes when applied to medium-sized industrial data sets. Model evaluation is done with the ROC curve.

Chapter 6 summarizes the conclusion and future work related to this study.

Chapter 2

LITRATURE SURVEY

With the emerging demand of change and new releases of softwares quite good amount of work has been done in the field of change prediction .Sharafat et al. [11] proposed a probabilistic model that was based on the change history as well as source code of the software. Reverse engineering technique was used to analyze the source code over the various releases of the software and various code metrics was collected. The probability of change was predicted on the basis of code metrics. For this study he had used medium sized system Flex , a lexical analyzer generator of java. Paper [12] conducted a study on three large softwares and validated CK[9] metrics and found that WMC and RFC plays a good role in the prediction of change prone classes .Zhou et al. [13] investigated the confounding effect of the class size on the relationship among the change proneness and the object oriented metrics by studying three size metrics namely number of methods implemented in a class (NIMMP), source lines of code (SLOC) and NumPara that is sum of number of parameters of the methods implemented in a class. Data set was collected from a java based software Eclips. He had found that class size has much impact on the association among the change proneness and object oriented metrics.

Chaumum [14] studied a C++ system and defined a model based on change impact and observed the impact of changes in the classes of the software. Change was mainly dependent upon the two main factors namely the type of impacted classes and the links between them

such as inheritance, aggregation, generalization. Tsantalis et al. [15] studied the impact of adding or modifying a class on the change proneness of the object oriented system. Two open source multiversion projects, JFlex and JMol were used for this study. Statistical analysis given improved results for the correlation among the extracted probabilities and the actual change in each class as compared to the model that was based on the past data. In paper [19] five systems were analyzed to get the relationship among the design patterns and change proneness .In one system classes were found more change prone while in rest if the four systems classes were found less change prone. The study guides us in learning the software design pattern that are easily adaptable. Foutse Khomh[16] performed study on the ten releases of ArgoUML ,18 of Mylyn, to investigate the relationship among the object oriented metrics and change-proneness.

Paper [24] studied UML 2.0 model for improving the design quality by predicting change proneness and he have done this by BDM (behaviour dependency measurement) that is a method which does the rating of classes by analyzing the probability of change. An open source software JFreeCharge was used for the analysis and evaluation of results .The author had concluded that BDM is quite effective indicator for prediction of change proneness. Ambros et al. [18] studied the relationship among the change coupling and the software defects by using correlation and regression analysis. Change coupling is defined as the interdependence of various artifacts of the software on each other. A change in one part may lead to several changes in the other part of the software .Three large sized systems were analyzed for this study. It was concluded that there is a correlation among the change coupling and defects .

Another paper by Briand et al. [24] has also validated the 49 metrics. The system used for this study was the multi-agent development system, which consists of three classes. They found NOC metric to be insignificant, while DIT was found to be significant in an inverse manner. WMC, RFC, and CBO were found to be strongly significant.

From the survey we have conducted, the following observations were made:

• We observed that among the number of metrics available in literature, the CK metric suite is most widely used. It has been seen that most of the studies have also defined their own metric suite and they have used them for carrying out the analysis.

• Among the various categories of methods available to predict the most accurate model such as machine learning methods, statistical methods, etc. the trend is shifting from the traditional statistical methods to the machine learning methods. It has been observed that machine learning is widely used in new bodies of research to predict change prone classes. Results of various studies also show that better results are obtained with machine learning as compared to statistical methods.

• Papers have used different types of datasets, which are mostly public datasets, commercial datasets, open source, or students/university datasets. We have observed that the public datasets, which have been mostly used in the studies, are from the PROMISE and NASA repositories.

Chapter 3

RESEARCH BACKGROUND

In this chapter we will discuss what change proneness is and follows it up with description of change proneness we used in study. Second, we will briefly describe about the object oriented metrics we used in our study. Third we talk about the empirical data collection for this study.

3.1 Change Proneness

During software evolution series of changes are made to software. Changes can be due to a variety of reasons such as enhancements, adaptation, perfective maintenance or fixing defects. Thus, change proneness is the probability that a system would undergo a change in its successive versions.

The study carried out by us revolves around finding a relationship between change prone classes of a system with object oriented metrics.

3.2 Metrics Selected for Study

The class-level metrics are used to indicate the internal properties of a class (LOC, NOM, etc) and the association between classes (CBO, NOC, etc).

We calculated metrics under different dimensions of object oriented systems viz. Inheritance, coupling, size and cohesion using Understand software tool. Understand is a static analysis tool for maintaining, measuring, & analyzing critical or large code bases. By using

Understand we can calculate up to 22 complexity metrics, 42 count metrics and 29 object-oriented metrics [43].

Out of these 93 metrics only 13 are required to predict a software system for the change proneness described above. Table 3.1 summarizes the metrics selected for this study and is followed by a brief description of each metric.

3.3 Independent and Dependant variable

This section describes about the variables and the metrics that we have used .In this paper, we have used change proneness as dependent variables. Change proneness is the probability of occurrence of the change after the release of the software [3] . Since software goes on with a lot of changes with the change in requirement, technology, and functionalities that's why changes are becoming quite common and important for now a days software's and hence change proneness factor is of keen importance for us to study investigate and find some relationship with the object oriented metrics in order to improve some quality factors and make our maintenance easier as well.

Object oriented metrics are used as Independent variable that are used throughout the software process and for this purpose a single metric is not sufficient and hence several metrics must be used to achieve our required goal.

So for this purpose we have used most commonly used metrics and the values of these metrics are got through understand for java tool ((http://www.scitools.com/) that we have explained in next section. Table 1 contains the brief description about the metrics that are used.

Table 3.1 List of metrics

S.No	Metric Name	Definition	Source
1	SLOC-Sorce lines of code	Number of lines in the source code	http://www.scitools.com/features/metrics.php
2	RFC-Response of a class	Sum of number of methods called within the class and methods accessible to that object because of inheritance	[9,2,4]
3	CBO-Coupling between the objects	number of classes to which it is coupled and vice versa	[4]
4	NOC- Number of children	Number of immediate subclasses of a class	http://www.scitools.com/features/metrics.php
5	NPRM-Number of private methods	Number of local(not inherited) private methods	http://www.scitools.com/features/metrics.php
6	LCOM-Lack of cohesion	For each data field in a class, the percentage of the methods in the class using that data	[9,2,4]

		field; the percentages are averaged then subtracted from 100 %	
7	WMC-Weighted methods per class	It is a count of sum of complexities of all methods in a class	[9,5,4]
8	BLOC-Blank lines of code	BLOC is the number of blank lines of code	http://www.scitools.com/features/metrics.php
9	NPM - Number of Public Methods	counts all the public methods in a class	[10]
10	NPROM-Number of protected methods	It counts number of local protected methods of a class	http://www.scitools.com/features/metrics.php
11	NLM-Number of local methods	NLM is the number of local methods of a class that are not inherited	http://www.scitools.com/features/metrics.php
12	SC-Statement	SC is the total number	ttp://www.scitools.com/features/metrics.p

	count	of declarative and executable statements	hp
13	CC - McCabe's Cyclomatic Complexity	It is the total number of independent paths in the flow graph of the program.	[7]
14	NL-Number of lines	NL is the total number of lines in the program	ttp://www.scitools.com/features/metrics.php
15	CBM-Coupling Between Methods	measures total number of new/redefined methods to which all the inherited methods are coupled.	[7]

3.2.1 Source lines of code (SLOC)

The number of lines that contain source code. A line can contain source and a comment and thus count towards multiple metrics. For Classes this is the sum of the source lines of code for the member functions of the class.

3.2.2 Response for a class (RFC)

The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class. If a large number of methods can be invoked in response to a message, the testing and debugging of the class becomes more complicated since it requires a greater level of understanding required on the part of the tester. The larger the number of methods that can be invoked from a class, the greater the complexity of the class. A worst case value for possible responses will assist in appropriate allocation of testing time [9].

3.2.3 Coupling between objects (CBO)

CBO for a class is a count of the number of other classes to which it is coupled. Excessive coupling between object classes is detrimental to modular design and prevents reuse. The more independent a class is, the easier it is to reuse it in another application. A measure of coupling is useful to determine how complex the testing of various parts of a design is likely to be. The higher the inter-object class coupling, the more rigorous the testing needs to be [9].

3.2.4 Number of children (NOC)

It is defined as number of immediate subclasses subordinated to a class in the class hierarchy. Greater the number of children, greater the reuse, since inheritance is a form of reuse. Greater the number of children greater the likelihood of improper abstraction of the parent class. If a class has a large number of children, it may be a case of misuse of subclassing. The number of children gives an idea of the potential influence a class has on the design. If a class has a large number of children, it may require more testing of the methods in that class [9].

3.2.5 Number of private methods (NPRM)

It gives the count of private methods defined in a class without considering the inherited methods.

3.2.6 Lack of cohesion (LCOM)

The LCOM is a count of the number of method pairs whose similarity is 0 minus the count of method pairs whose similarity is not zero. The larger the number of similar methods, the more cohesive the class, which is consistent with traditional notions of cohesion that measure the inter-relatedness between portions of a program [9].

3.2.7 Weighted methods per class (WMC)

The WMC metric is the sum of the complexities of all class methods. It is an indicator of how much effort is required to develop and maintain a particular class. A class with a low WMC usually points to greater polymorphism. A class with a high WMC indicates that the class is complex (application specific) and therefore harder to reuse and maintain [9].

3.2.8 Number of Blank lines (BLOC)

BLOC is the number of blank lines of code

3.2.9 Number of public methods (NPM)

It stands for number of public methods of a class that are defined locally i.e. the methods which are not inherited from a base class.

3.2.10 Number of protected methods (NPROM)

For a class, count of protected methods declared in it is NPROM. Protected methods that are inherited are generally discarded in this count

3.2.11 Number of local methods (NLM)

NLM is the number of local methods of a class that are not inherited

3.2.12 Statement count (SC)

SC is the total number of declarative and executable statements

3.2.13 McCabe's Cyclomatic Complexity (CC)

McCabes's complexity is a flow based metric and it is defined as the total number of independent paths in the flow graph of the program .So many times this metric plays a very important role in the prediction og the change proneness since it is related to the flow of the program and hence any change in this can have very large impact on the whole program.

3.2.14 Number of Lines (NL)

NL is the volume based metric and it tells us the total number of lines in the program so many times it is seen that the size of the program also have some influence in the pdediction of the change proneness but right now we are not considering size of the program as an influencing factor but as far as our future study is considered we will take also size of the program as a factor in the prediction of the change proneness.

3.2.15 coupling between the methods (DIT)

Coupling between the methods is a quite important metrics that plays a very vital role in the prediction of change proneness and it is defined as the measures total number of new/redefined methods to which all the inherited methods are coupled.

Chapter 4

RESEARCH METHODOLOGY

In this chapter we have explained about the 10 machine learning techniques that we have used for the prediction of change proneness along with the brief description of data collection and analysis of data as well.

4.1 Empirical Data Collection

For this study we have used two open source data set “J-Meter” and “ XML-Security”[8].

Table 4.1 Summary of Dataset used

Name	V1	V2	P/L used	Total LOC	Total classes	Classes exhibiting change	Classes without change
J-Meter	2.8	2.9	Java	43400	335	135	335
XML-security	1.0	1.1	Java	74762	159	37	122

4.1.1 Statics for J-Meter

J-Meter is a 100 percent pure java API which is designed to load and test performances and functional behaviour of the system. It was designed for testing of the web applications but

now it is expanded to test so many functions. J-Meter may be used to test the performance of static as well as dynamic resources and it can also be used to simulate the heavy load on the network, server or object and can test the strength. It can also analyze the performance under different types of load. System consists of 335 classes with 43400 lines of code. We have taken two versions namely Jmeter2.8 and Jmeter2.9 and found the number of lines added deleted and modified in the respective classes of the two versions .The details of the changes in the classes can be seen in the figure-1 given below. We have defined a change factor that is

Change factor =number of lines added + number of lines deleted +number of lines modified*2

Table two contains the details of change factor that the classes are gone though and we can analyze the 176 classes are gone through CF of range 1 to 10 that are almost 52 percent of classes are lying in this range,.

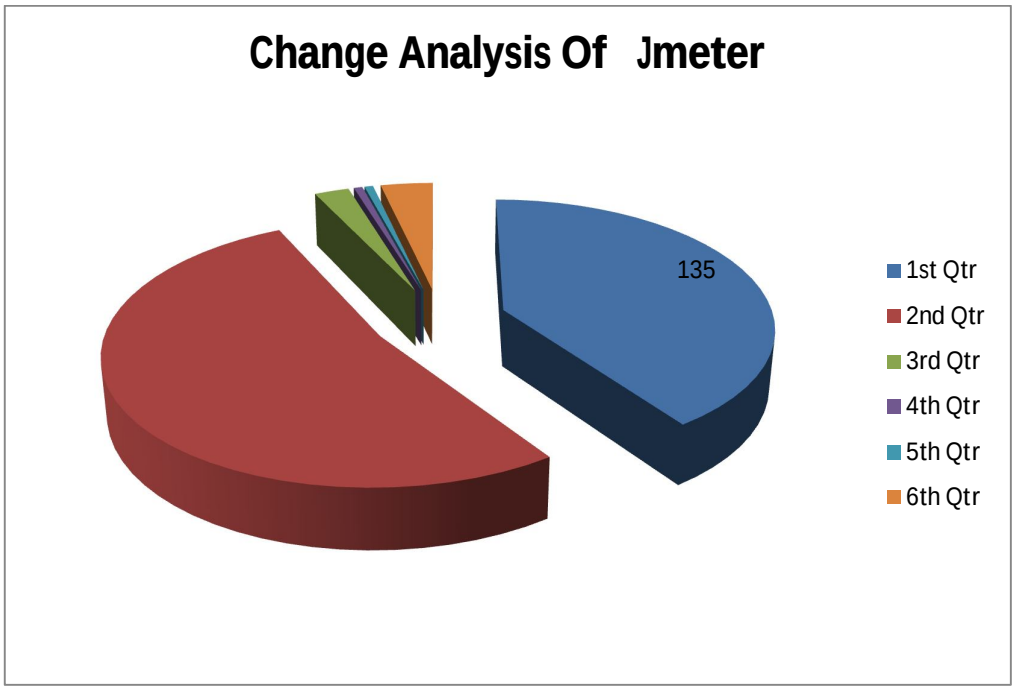


Fig-4.1 change analysis of J-Meter

We can visualize about the changes in the fig 4.1 that out of 335 classes 135 classes don't have any change that is 40 percent classes remains as is it without any change . 2nd quarter is the largest one that is showing a change between 1 to 10 means this is the most critical range and almost 52 percent of classes are gone through this much amount of change factor in 3rd quarter is that is showing a range of 20-30 change factor is covering 2 percent . 4th and 5th quarters are showing a very small change that is only 2 classes gone through this much amount of change.

Table-4.2 Descriptions of changes for J-meter

Range of Change Factor	Number of classes	Percentage
No change	135	40.29
1-10	176	52.53
11-20	8	2.38
21-30	2	.05
30-40	2	.05
40 above	12	3.58

Table -4.3 Descriptive Statistics of J-Meter

Metric	Min	Max	Median	Mean	Standard deviation
SC	2	596	29	58.44	81.15
LC	0	557	16	37.49	63.88
BLOC	0	182	9	16.49	22.41
NL	3	1304	70	132.94	181.26
NPM	0	18	0	1.21	2.47
NPRM	0	102	4	6.94	9.87
NPROM	0	19	0	.53	1.83
NLM	0	112	5	8.91	11.39
CBO	0	34	2	3.67	4.53
NOC	0	17	0	.33	1.56

LOC	3	876	44	84.41	113.85
DIT	1	5	2	1.83	.918
LCOM	0	100	68	60.21	33.61
WMC	0	152	9	16.63	22.90
NIM	0	108	4	7.83	10.84
NIV	0	42	1	2.41	4.63
RFC	0	121	8	19.87	25.16

4.1.2 Statics for XML-Security:

XML-security is a pure java API and this is used for implementing XML signature and encryption standards, supplied by the XML subproject for the open source Apache project.XML security have 159 classes and 16800 lines of code.

Table 4.4 Description of changes for XML-security

Range of Change Factor	Number of classes	Percentage
No change	37	23.27
1-10	59	37.10
11-20	29	18.23
21-30	8	5.03
30-40	4	2.51
40 above	30	18.86

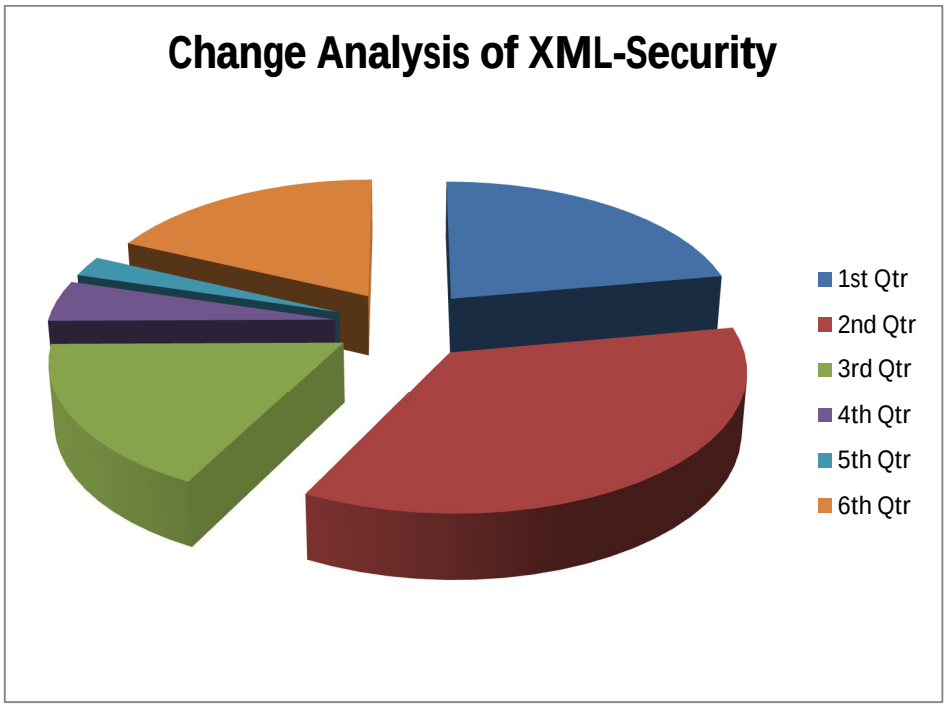


Fig-4.2 change analysis of XML-security

For XML-security 58 classes out of 159 are gone through a change factor between 1 to 10 means around 37 percent classes are gone through this much amount of change. 37 classes don't have any change and 29 classes have the change in between 20 to 30.

Table -4.5 Descriptive Statistics of XML-security

Metric	Min	Max	Median	Mean	Std Dev
SC	2	352	35	58.99	60.09
LC	0	635	15	36.32	60.32
BLOC	1	190	20	32.59	37.64

NL	9	1021	194.64	208.81	120
NPM	0	8	0	.47	1.12
NPRM	0	53	5	7.79	7.88
NPROM	0	19	0	.64	.72
NLM	0	124	8	9.12	17.39
CBO	0	22	3	4.47	4.44
NOC	0	24	0	.78	2.69
LOC	3	552	58	95.96	107.93
DIT	1	4	2	2.20	.87
LCOM	0	100	65	50.49	40.18
WMC	0	124	12	18.89	21.49
NIM	0	105	5	8.65	11.21
NIV	0	35	3	6.52	9.65
RFC	0	89	21	23.54	15.96

4.2 Calculation of changes in the two different versions

We have calculated number of lines added, deleted, and modified keeping few points in mind as follows

- Comments are completely discarded means if there are any comments added in the new version then it not counted in number of lines added and if any comment is not available in new version then it is also not considered as deleted line.

- Import statements are also not considered.
- Addition of any new line in the new version is counted in added line
- Any source code line which is present in the older version but not present in the new version is treated as deleted line.
- Blank lines are also not considered in added, deleted or modified lines.
- Lines which are present in the older version but it was commented in the new version is treated as deleted line.

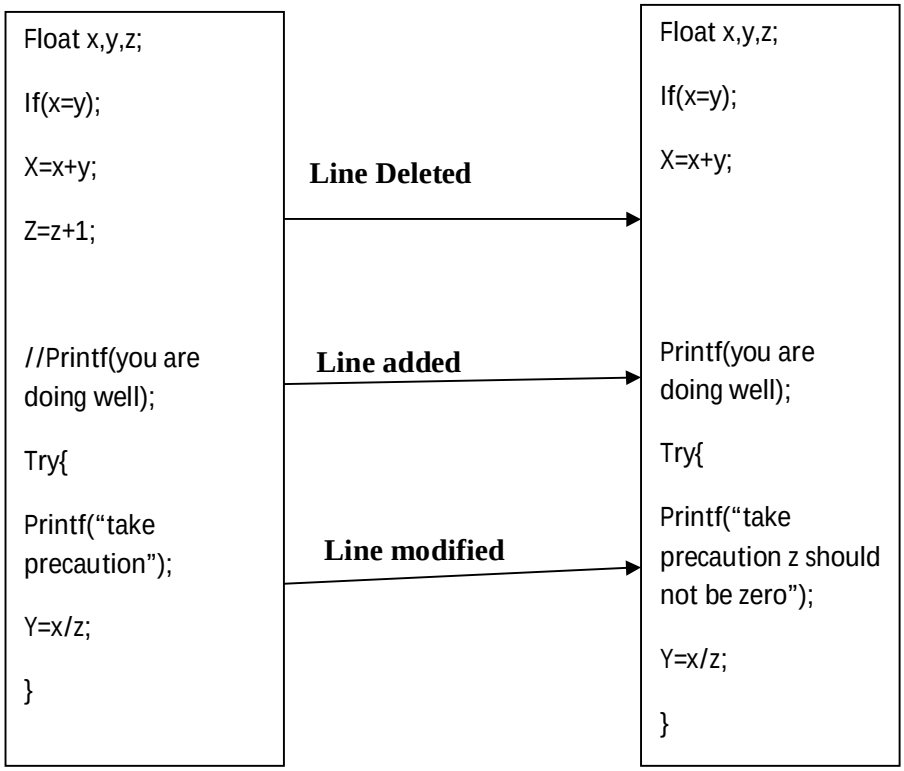


Fig. 4.3 Updation rule

After calculating these changes we have defined a binary variable “OVERALL CHANGE” and if the value of changes is greater than 0 it is treated as “yes” and if it is zero then it is treated as “no”;

4.3 Analysis of data for change prone classes.

Now after the metrics have been calculated with the help of understand for java tool we have calculated number of lines added, deleted and modified with our own developed tool that is being validated by testing in so many projects. After this empirical data collection we have got all those classes in which changes have taken place and then we have applied machine learning techniques to analyze the change prone classes and also found the area under the curve of ROC that is seems quite good.

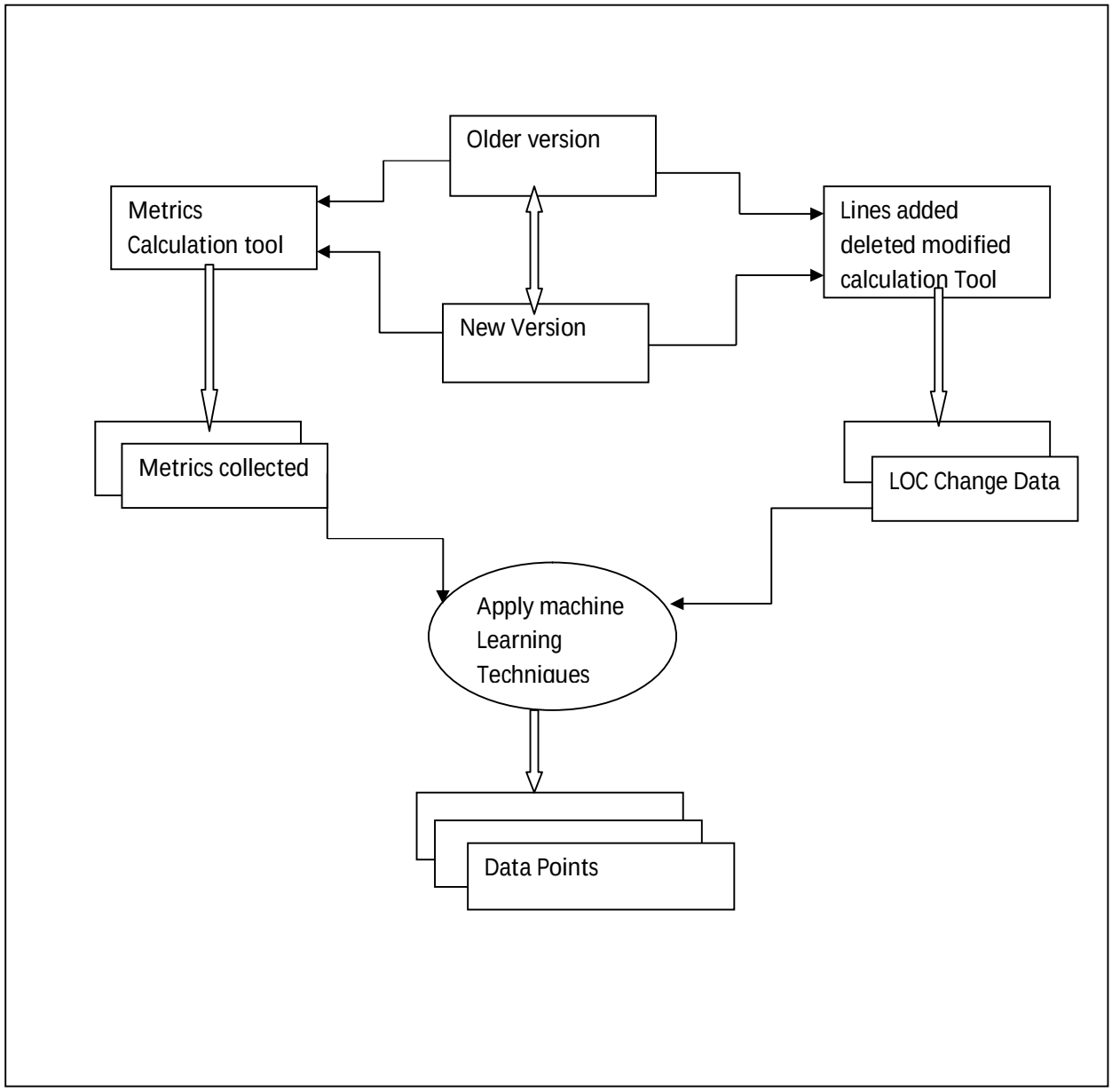


Fig-4.4 Data Analysis Life cycle

4.4 Methods Used:

In this section we will discuss a little about the methods that we have used for the prediction of change prone models. In order to predict the change proneness of classes, we have used the WEKA open source tool [35].

4.3.1 The statistical model

Logistic regression is the commonly used statistical modeling method. Logistic regression is

Used to predict the dependent variable from a set of independent variables (a detailed description is given by [3, 31, 32]). It is used when the outcome variable is binary or dichotomous. We have used both univariate and multivariate regression. Univariate logistic regression finds the relationship between the dependent variable and each independent variable any significant association between them. Multivariate logistic regression is done to construct a prediction model for the change proneness of classes. It analyzes which metrics are useful when they are used in combination. Logistic regression results in a subset of metrics that have significant parameters. To find the optimal set of independent variables (metrics), there are two stepwise selection methods, which are forward selection and backward elimination [32]. Forward selection examines the variables that are selected one at a time for entry at each step. The backward elimination method includes all the independent variables in the model and the variables are deleted one at a time from the model until the stopping criteria is fulfilled. We have used the forward stepwise selection method.

The general multivariate logistic regression formula is as follows [3]:

$$\text{Prob } (X1, X2, \ldots, Xn) = \frac{e^{g(x)}}{1 + e^{g(x)}}$$

where g(x) = B0 + B1*X1 + B2* X2 + … + Bn* Xn

'prob' is the probability of a class being change prone.

Xi, ($1 \leq i \leq n$) are independent variables

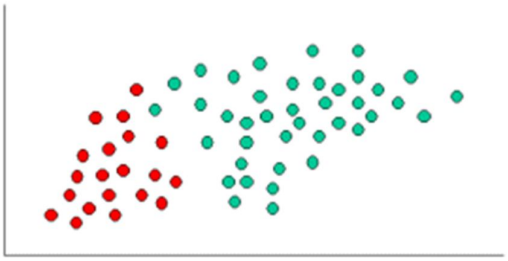
The following statistics are reported for each metric from the above formula:

1. Odds Ratio: The odds ratio is calculated using Bi's. The formula for the odds ratio is R= exp (Bi). This is calculated for each independent variable. The odds ratio is the probability of the event divided by the probability of a non-event. The event in our study is the probability of having a change and the non- event is the probability of not having a change [4].

2. Maximum Likelihood Estimation (MLE) and coefficients (Bi's): MLE is the likelihood function that measures the probability of observing a set of dependent variables [4]. MLE finds the coefficient in such a way that the log of the likelihood function is as large as possible.

4.4.1 Naïve Baiyes: Naïve bays classification method is based on the Bays rule and is relies on simple representation of documents [*paper naïve bays*]. It is based on the Bayesian theorem and particularly suited when the dimensionality of the inputs is high. Parameter

estimation for this model uses the method of maximum likelihood [lab 4 naïve bays]. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.



To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects [20]. There are twice as many GREEN objects as RED, so it is reasonable to believe that a new case is twice probability to have membership GREEN rather than RED. This belief is known as the prior probability in Bayesian Belief. Prior probabilities are based on past experience, so in this case the percentage of GREEN and RED objects, often used to predict outcomes before they happen.

4.4.2 Bayes Net:

Bayesian networks pearl (1988) are quite powerful probabilistic representation and that's why they are most often used for classification purpose but unfortunately they perform in a poor way when learned in a standard way[22].Bayes Nets are graphical representation for probabilistic relationships among a set of random variables. Given a finite set X(x1,x1,x3…….xn) of discrete random variables where each variable Xi may take values from a finite set, denoted by Val(Xi)[21]. A Bayes net is an annotated DAG(directed acyclic

graph) G that encodes a joint probability distribution over X. In the Bayes networks The nodes of the graph correspond to the random variables X1,X2……...Xn The links of the graph correspond to the direct influence from one variable to the other. If there is a directed link from variable Xi to variable Xj, variable Xi will be a parent of variable Xj. Each node is annotated with a conditional probability distribution (CPD) that represents p(Xi/Pa(Xi)), where Pa(Xi) denotes the parents of Xi in G. The pair (G, CPD) encodes the joint distribution p(Xi,…….Xn).

4.4.3 J48:

J48 is an open implantation c4.8 algorithm by weka tool in java and this is decision tree based algorithm that builds the tree in the same way as ID3 along with some improvements. Ros quinlan had developed this algorithm and this is now widely used for the classification purpose now a days. In this algorithm first base cases are checked and then for each attribute normalized information gain is found and the attribute that has the highest information gain is made the root node and this process is done recursively [23] . J48 is an evolution and refinement of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on makes it more fruitful.

4.4.4 Random forest:

Random forest is quite popular and versatile machine learning classification algorithm and it can work on many attributes with large datasets. Beside the class tags it can also provide some other important information about the dataset. It consists of bagging of un-pruned decision tree learners with randomized features at each split [24] .Decision trees are the most

commonly method used for the data exploration such as CART and regression trees. The forest consists of randomly selected inputs or combination of inputs at each node to grow each tree. Random forest is simple and relatively robust to noise and gives quite good result for some data sets with fast learning. Accuracy of Random forest is as good as Adaboost and sometimes it also gives better result than this. One more advantage of this algorithm is that it is relatively faster than the bagging with better strength, variable importance and correlation.

4.4.5 Bagging

Bootstrap aggregation is known as bagging and proposed by Breiman[25].It is based on the uniformly and randomly collecting n samples with the replacement of n size sample[27] It constructs of many different bags of samples and by performing bootstrapping iteratively, Classification of each bag is done , and computing some type of an average of the classifications of each sample via a vote. This algorithm is in some ways similar to boosting, because both methods design a collection of classifiers and combine their conclusions with a vote. Bagging also uses resampling than reweighting and does not change the sample distribution .The idea consists of using multiple versions of a training set; each version is created by randomly selecting samples of the training dataset, with replacement, where n is the number of training samples of the original training set [26]. In this a regression model is build by applying previously selected algorithms [26]. In this various similar training sets are build and a new function for each of them is trained. And when we predict numerical result t his accumulation standardizes over the versions. [3]

4.4.6 nnge

Nearest-neighbor-like algorithm using non-nested generalized exemplars (which are hyperrectangles that can be viewed as if-then rules).

4.4.7 Conjunctive Rule:

Conjunctive Rule is this class implements a single conjunctive rule learner that can predict for numeric and nominal class labels (Platt,1998).Single conjunctive rule learner a popular machine learning algorithm and is also normally known as inductive learning. The goal of rule induction is to induce a set of rules from the given data that captures all generalizable knowledge within data, and at the same time being as small as possible [10]. Classification in rule-induction classifiers is based on the firing of a rule on a test instance and triggered by matching feature values at the left side of the rule [11]. Rules in this algorithm can be of various normal forms, and are typically ordered; with ordered rules, the first rule that fires determines the classification outcome and halts the classification process. Conjunctive descriptions are easily learned by finding all commonalities shared by all positive examples [29].

4.4.8 K Star:

K* is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function[30]. It differs from other instance-based learners in that it uses an entropy-based distance function. It uses such a measure, and results are presented which compare favourably with several machine learning algorithms .Instance-based learners classify an instance by comparing it to a

database of pre-classified examples. The fundamental assumption is that similar instances will have similar classifications [30].

Chapter 5

RESULT ANALYSIS

In this section we have discussed the result that we have got by applying various machine learning and statistical techniques .CK metric suit is validated in this study. First of all we have identified the subset of metrics that are related to change proneness .In statistical modelling we have used multivariate logistic regression. After that we have applied multivariate logistic regression technique that can be used to predict overall change in the system and in order to predict the best model we have used various machine learning technique

Now in order to measure the performance of the predicted model, we have used the following performance evaluation measures

Sensitivity: It is defined as the percentage of classes that are correctly predicted to be change prone

Sensitivity = ((Number of classes correctly predicted as change prone) / (total number of actual changed classes)) * 100

Specificity: It also measures the correctness of the predicted model. It is defined as the percentage of classes predicted that will not be change prone [33].

Specificity = ((Number of classes correctly predicted as non- change prone) / (total number of actual non changed classes)) * 100

Precision or Accuracy: It is defined as the ratio of number of classes (including change prone and not change prone) that are predicted correctly to the total number of classes [33].

Receiver Operating Characteristic (ROC) analysis: We have used ROC in order to evaluate the performance of the outputs of the predicted models. It is seen that this is an effective method of evaluating the performance of the model predicted. The ROC curve is defined as a plot of sensitivity on the y coordinate versus its specificity on the x-coordinate [31]. While constructing ROC curves, we selected many cut-off points between 0 and 1, [33] and calculated sensitivity and specificity at each cut-off point. The ROC curve is used to obtain the required optimal cut-off point that maximizes both sensitivity and specificity [31, 32].

The *validation method* used in our study is k-cross validation (the value of k is taken as 10) in which the dataset is divided into approximately equal k partitions [34]. One partition at a time is used for testing the model and the remaining k-1 partitions are used for training the model. This is repeated for all the k partitions.

5.1 Univariate LR Analysis Results

We conducted univariate analysis to find whether each of the metrics (independent variables) is significantly associated with change proneness (dependent variable). Table 5.1 and Table 5.2 represents the results of univariate analysis. It provides the coefficient (B), standard error (SE), statistical significance (Sig.), and odds ratio (Exp (B)) for each metric [21]. The parameter "sig" tells whether each of the metric is a significant predictor of change proneness. If the "sig" value of a metric is below or at the significance threshold of 0.01, then the metric is said to be significant in predicting the change prone classes [21].

.

Table 5.1 Univariate Analysis of J-Meter

Independent Variable	**Coeff.(B)**	**S.E.**	**Sig.**	**Exp(β)**	**α**
NIB	*0.099*	*0.084*	*0.235*	*1.105*	*-0.876*
NOA	***0.12***	***0.02***	***0.001***	***1.032***	***-0.958***
CBO	***0.123***	***0.034***	***0***	***1.131***	***-0.015***
NOC	***0.665***	***0.296***	***0.025***	***1.944***	***0.286***
WMC	*0.004*	*0.005*	*0.414*	*1.004*	*0.32*
NPM	*0.052*	*0.012*	***0.001***	*1.042*	*-1.555*
NPROM	***0.209***	***0.103***	***0.044***	***1.232***	***0.304***
NPRM	***0.254***	***0.055***	***0.021***	***1.202***	***-1.361***
LOC	*0.002*	*0.001*	*0.129*	*1.002*	*0.256*
DIT	*0.066*	*0.122*	*0.591*	*1.068*	*0.273*
LCOM	***0.007***	***0.003***	***0.035***	***1.007***	***-0.023***
RFC	***0.02***	***0.004***	***0.000***	***1.020***	***-1.398***

Table 5.2 Univariate Analysis of XML-security

Independent Variable	**Coeff.(B)**	**S.E.**	**Sig.**	**Exp(β)**	**α**
NIB	***0.657***	***0.237***	***0.005***	***1.929***	***-1.637***
NOA	***0.09***	***0.03***	***0.003***	***1.094***	***-0.984***
CBO	***0.282***	***0.064***	***0***	***1.326***	***-0.613***
NOC	*-0.099*	*0.07*	*0.16*	*0.906*	*0.515*
WMC	*0.018*	*0.011*	*0.093*	*1.018*	*0.019*
NPM	*0.033*	*0.047*	*0.489*	*1.033*	*0.354*
NPROM	***1.216***	***0.579***	***0.036***	***3.375***	***0.244***
NPROM	*1.216*	*0.579*	*0.036*	*3.375*	*0.244*
LOC	***0.005***	***0.002***	***0.012***	***1.005***	***0***
DIT	***-0.395***	***0.193***	***0.041***	***0.674***	***1.323***
LCOM	***0.035***	***0.005***	***0***	***1.035***	***-1.139***
RFC	***0.022***	***0.01***	***0.025***	***1.022***	***0.058***

Table 5.1 and Table 5.2 shows the significant values in bold. The coefficient "(B)" shows the strength of the independent variable. The higher the value, the higher the impact of the independent variable is. The sign of the coefficient tells whether the impact is positive or negative. We can see that NIB, WMC, DIT and LOC metrics for J-Meter and NOC, WMC, NPROM and NPM for XML-Security are not significant and are therefore not taken for any further analysis. Thus, in this way we can reduce the number of independent variables and select only the best change proneness predictors.

5.2 Multivariate Logistic Regression Analysis of J-Meter

Multivariate analysis is done to find the combined effect of all of the metrics together on change proneness. For doing multivariate analysis, we have used forward stepwise selection to determine which variables should be included in the multivariate model .Out of all the variables, one variable in turn is selected as the dependent variable and the remaining others are used as independent variables[35].

Table 5.3 Multivariate model statics for J-Meter.

Metric Name	B	SE	Sig	OR
CBO	.081	.039	.037	1.084
NOC	.972	.363	.007	2.642
DIT	.521	.175	.003	1.683
LCOM	.1000	.020	.010	1.532
RFC	-.002	.007	.001	.978
CONSTANT	-.3.388	.540	.000	.034

5.3 Multivariate Logistic Regression Analysis of XML-security

The multivariate analysis result of XML-security is provided in the table .The coeff (B), error (SE), statistical significance (Sig.), and odds ratio (Exp (B)) are also shown in the table for all the metrics included in the model and the result of machine learning techniques is given the table 5.2 .Only 5 metrics are (CBO, NOC, RFC, NPRM, DIT) are included in the model.

Table 5.4 Multivariate model statics for XML-Security

Metric Name	B	SE	Sig	OR
CBO	.328	.080	.000	1.388
NOC	-.194	.108	.072	.824
RFC	.089	.032	.005	1.093
NPRM	-.159	.050	.001	.853
DIT	-.1.910	.517	.000	.148
CONSTANT	-1.065	.312	.000	.034

5.4 Model evaluation using the ROC curve

In this section we will discuss in brief about the result evaluation of model using the RFC curve .We have used 10 machine learning technique to predict the accuracy of change proneness.

5.4.1 Evaluation for J-Meter

Table 5.5 Contains the result of cross validation of models predicted using various machine learning techniques .It have sensitivity, precision, area under curve, specificity, cut off point for the model that we have predicted using all the machine learning technique.ROC is used to find the cut off points and the cut off points are selected in such a way that shows the balance between the number of classes predicted as being change prone and not change prone.

Table 5.5 Validation Result for J-Meter

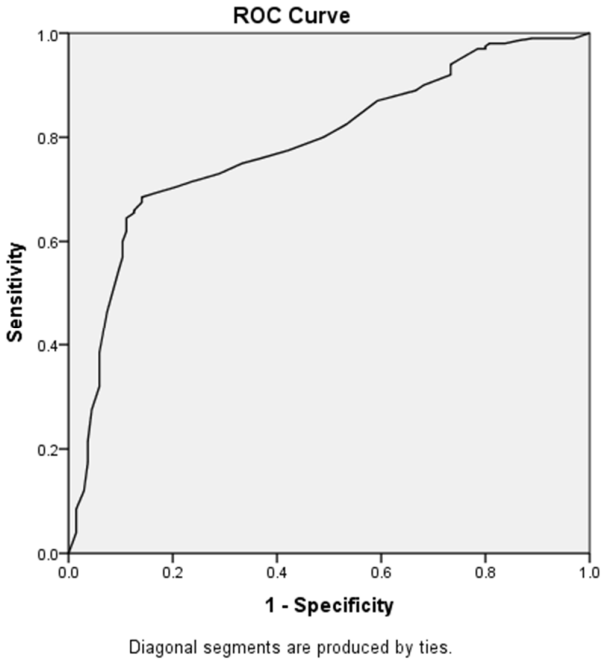
S.N	Method used	Area Under Curve	Sensitivity	Specificity	Cut-Off point
1	LR	.745	70.7	77.6	.603
2	Bayes Net	.786	73.0	71.1	.519
3	Naïve Bayes	.715	67.0	66.7	.133
4	J48	.800	81.5	79.3	.7085
5	Random forest	.884	82.5	83.0	.550
6	Adaboost	.818	75.0	74.1	.4925
7	Bagging	.833	84.0	83.7	.5545
8	Part	.840	82.0	81.5	.845
9	Conjunctive rule	.785	74.0	68.9	.3645
10	K star	.826	74.5	74.1	.5215
11	nnge	.820	85.5	78.5	.785

The y axis of ROC curve shows the sensitivity and the x axis shows the specificity and the point where the value of sensitivity becomes almost equal to the specificity is known as cut off point. ROC curves for the various machine models are shown in fig-4

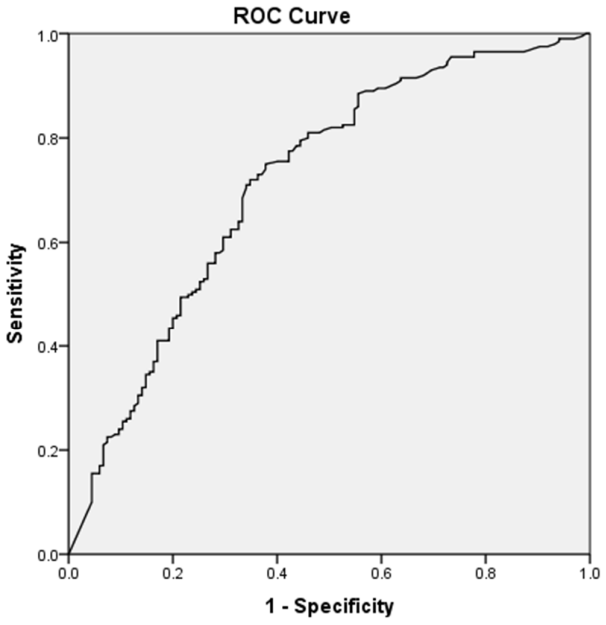
As we can see that here all machine learning models are showing quite better as compared to all the previous studies that have been done in this area also we can see that Area under curve for the Random forest is .884 and its specificity is around 83.0 percent that is really a good indication. Part is also giving quite goods result that is 82.0 percent sensitivity and .840 area under the curve that is quite near to bagging also we can see that AUC of conjunctive rule is

exactly same as that of Bays Net and apart from this conjunctive rule is also giving 74.0 percent sensitivity that is near to the sensitivity of adaboost. Bays net and Naïve bayes are also showing quite comparable results with .786 and .715 area under the curve but specificity of naïve bays is around 73.0 percent that is quite good as compared to Naïve Bayes which shows 67.0 percent only.j48 is a tree based learning and it also showing quite good results with 81.5 percent sensitivity and .800 area under the curve of ROC. Bagging is showing 83.36 areas under the curve with 84.0 sensitivity and 83.7 specificity that is minimal amongst all and hence can't be said good. K star and Naïve bayes are showing average results.

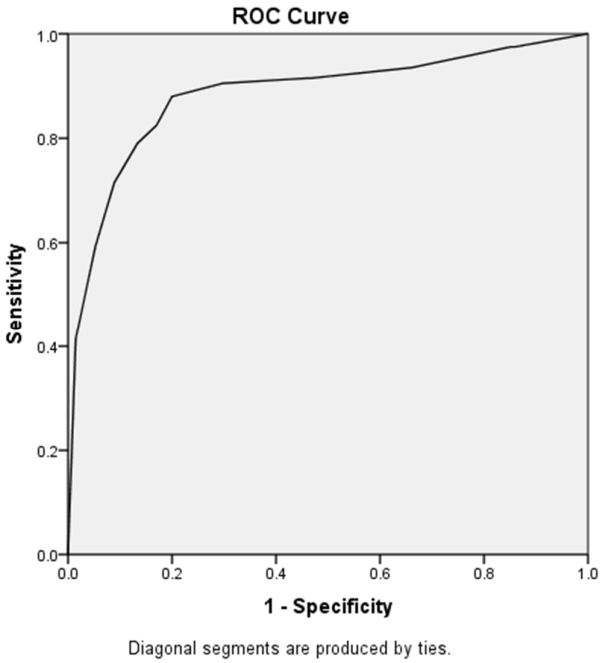
a) **Fig 5.1** Bayes Net



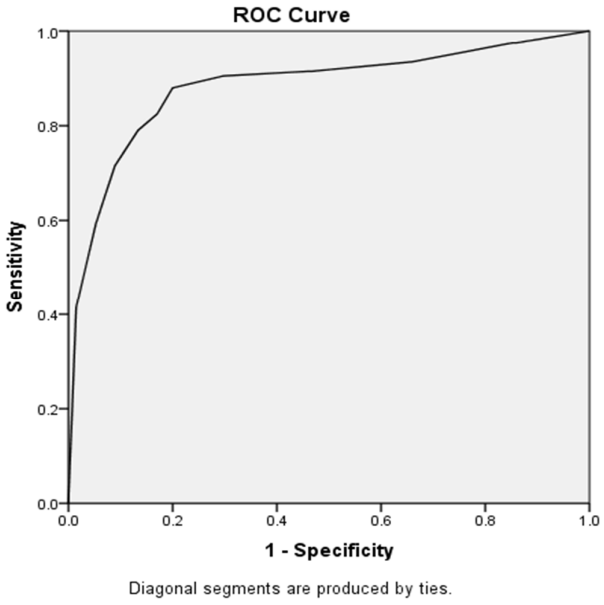
b) **Fig 5.2** Naïve Bayes



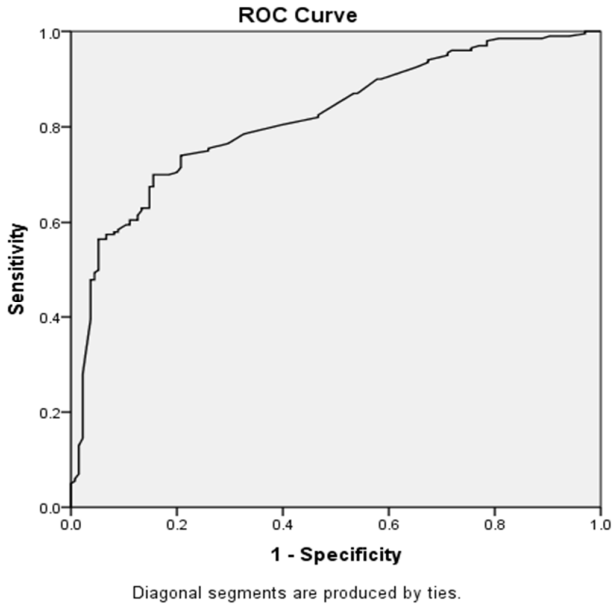
c) **Fig 5.3** J48



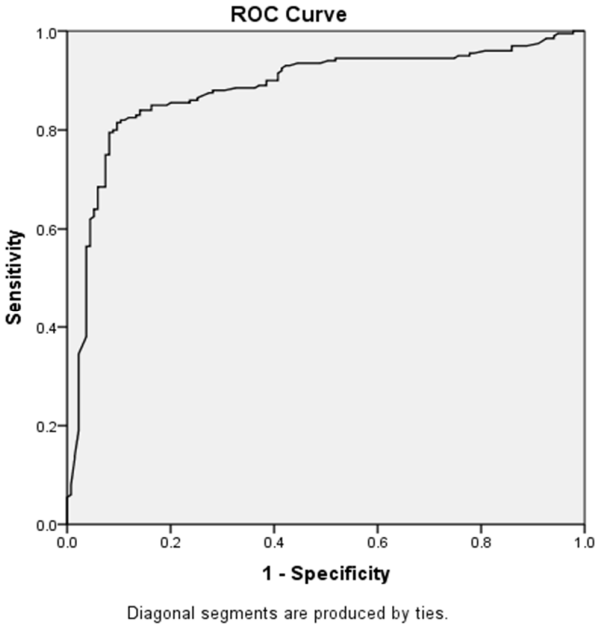
d) **Fig 5.4** Random forest



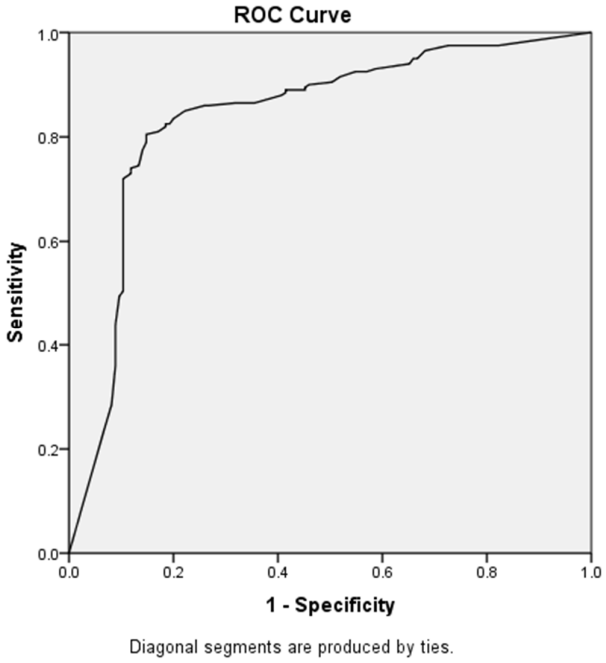
e) **Fig 5.5** Adaboost



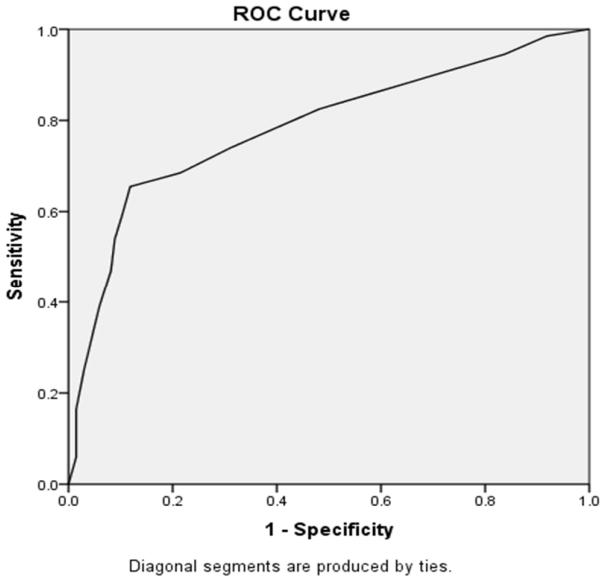
f) **Fig 5.6** Bagging



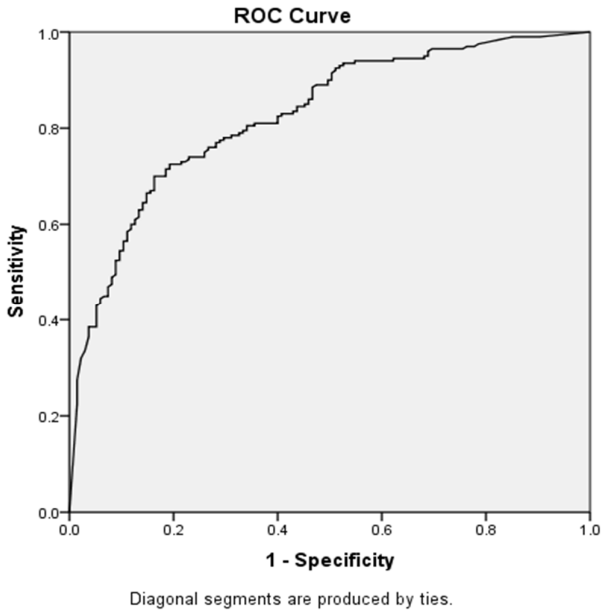
g) **Fig 5.7** Part



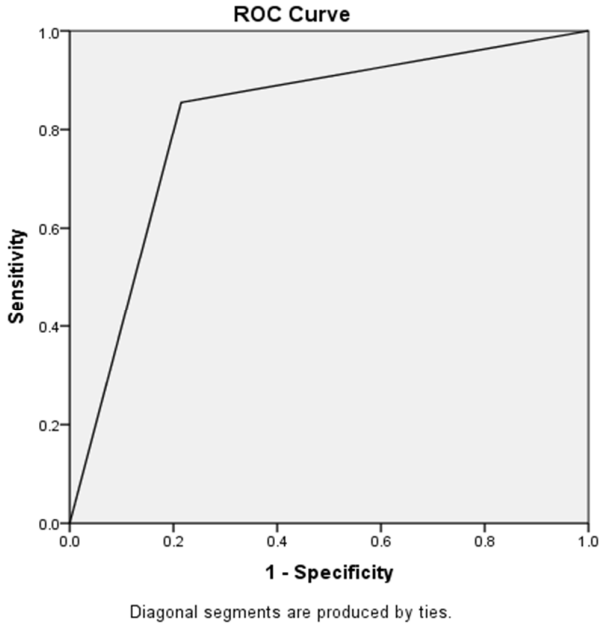
h) **Fig 5.8** Conjunctive rule



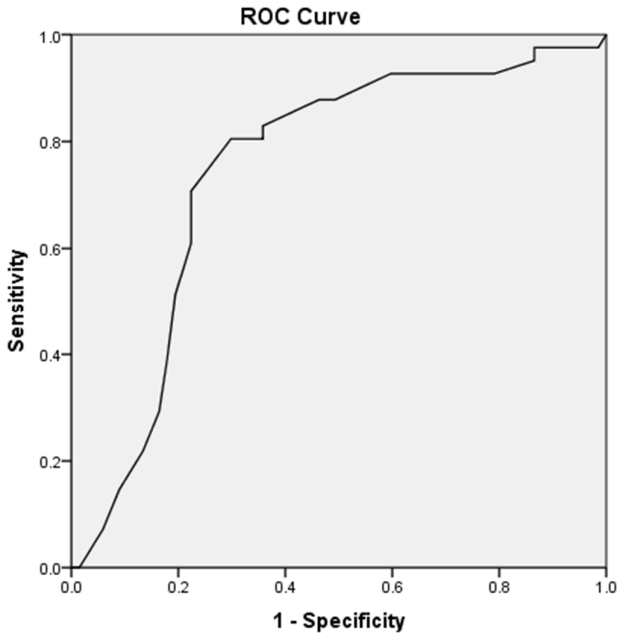
i) **Fig 5.9** K star



j) **Fig 5.10** nnge



k) **Fig 5.11** Logistic regression



As for as specificity is concerned Bays net is showing most fruitful result and the value of highest sensitivity is no doubt shown by the adaboost. Thus we can conclude that adaboost and Bayes net are the most appropriate machine learning model.

5.4.2 Evaluation for XML-security

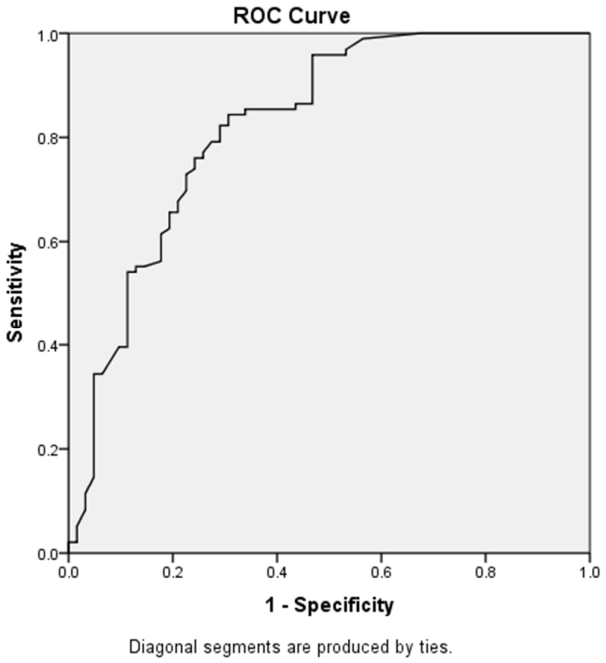
Table 4 contains the result of cross validation of models for the J-security software. Random forest again giving quite good result with .868 AUC and 77.1 percent sensitivity .K star and nnge are showing very competitive result and AUC for these are .837 and .839 respectively. Adaboost is showing highest accuracy with .877 AUC and 80.2 percent sensitivity that is exactly same as that of bagging.

Table 5.6 Validation Result for j-security

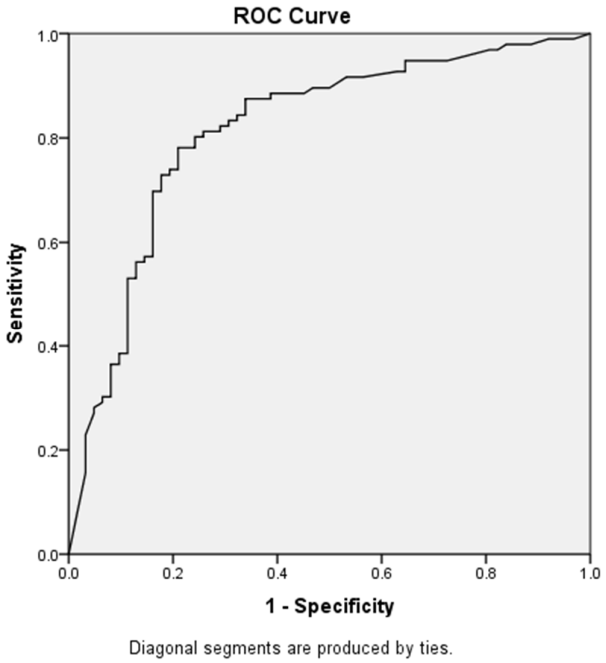
S.N	Method used	Area Under Curve	Sensitivity	Specificity	Cut-Off point
1	LR	.747	68.3	67.2	.515
2	Bayes Net	.824	76.0	75.8	.681
3	Naïve Bayes	.815	78.1	76.0	.315
4	J48	.841	80.2	79.0	.775
5	Random forest	.868	77.1	77.4	.65

6	Adaboost	.877	80.2	79.0	.569
7	Bagging	.856	80.2	80.6	.6565
8	Part	.799	72.9	79.0	.8865
9	Conjunctive rule	.789	78.1	74.2	.336
10	K star	.837	76.0	77.4	.823
11	nnge	.839	86.5	61.3	.5

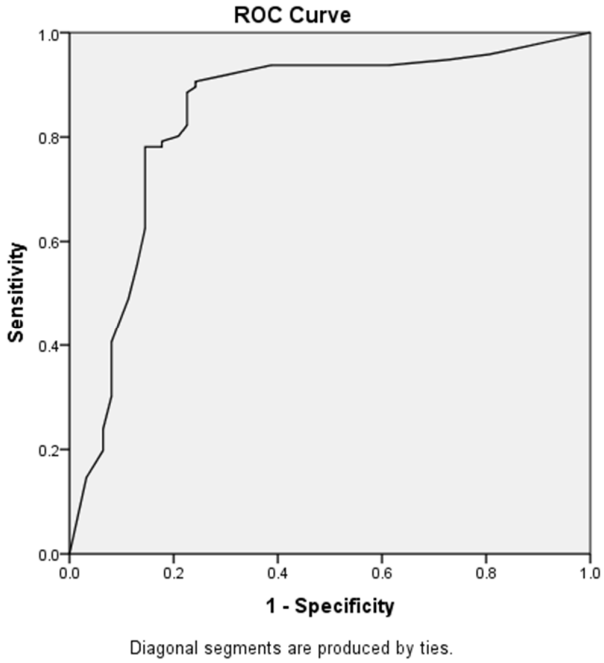
a) **Fig 5.12** Forward LR



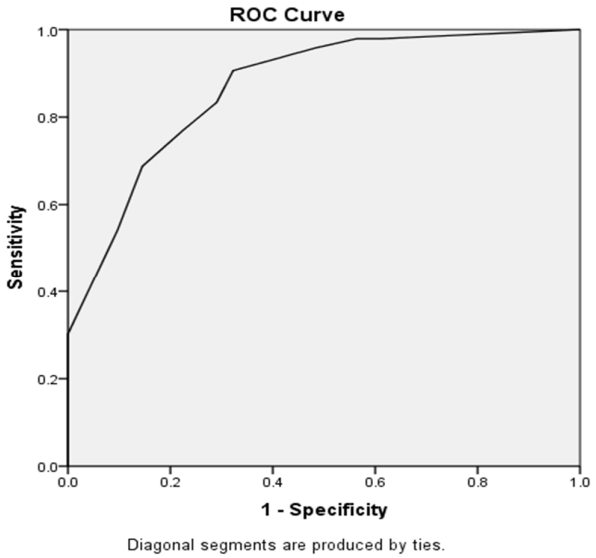
b) **Fig 5.13** Bayes Net



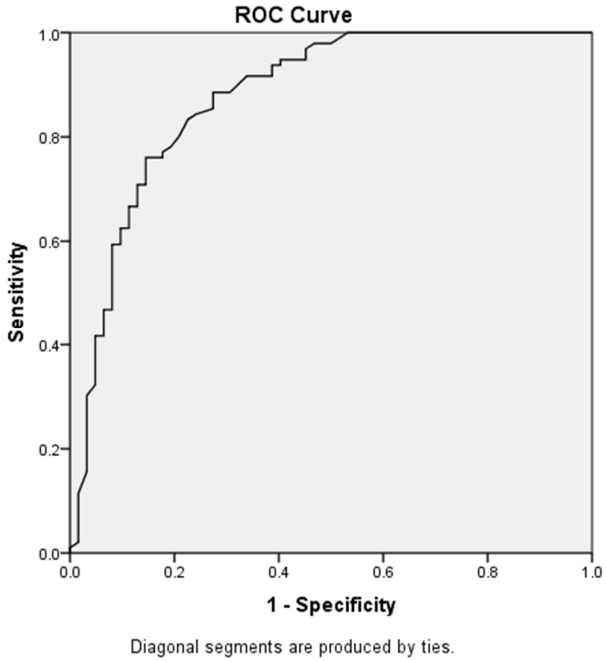
d) **Fig 5.14** Naïve Bayes



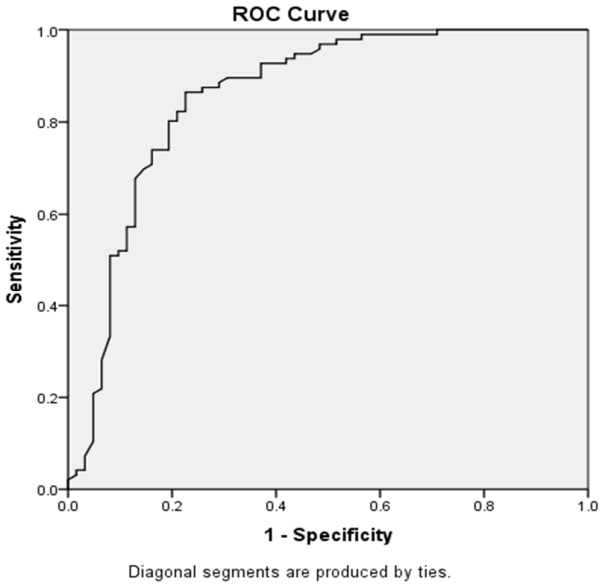
d) **Fig 5.15** J48



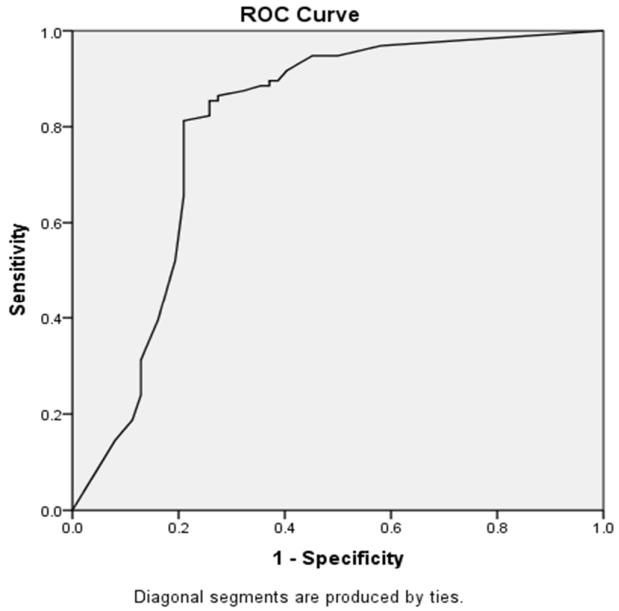
e) **Fig 5.16** Random forest



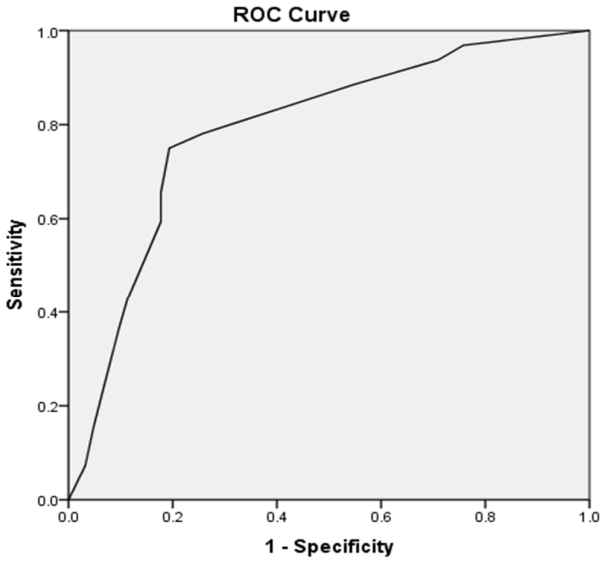
f) **Fig 5.17** Adaboost



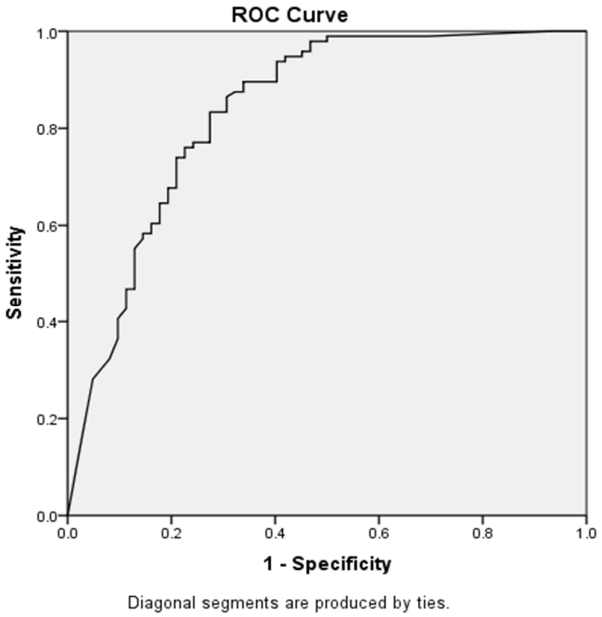
g) **Fig 5.18** Bagging



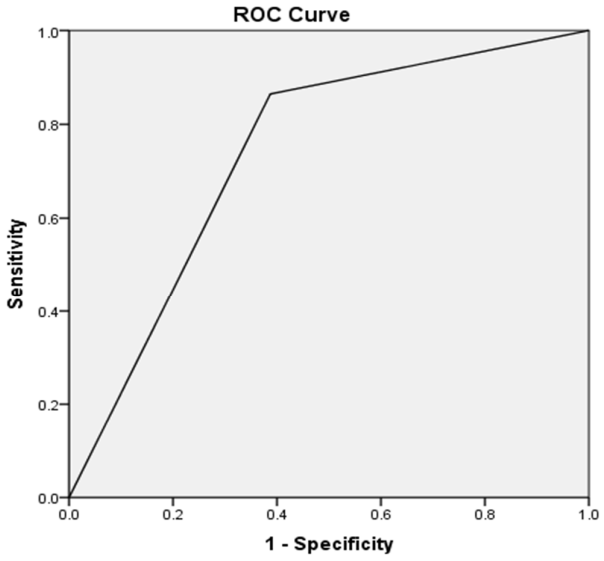
h) **Fig 5.19** Part



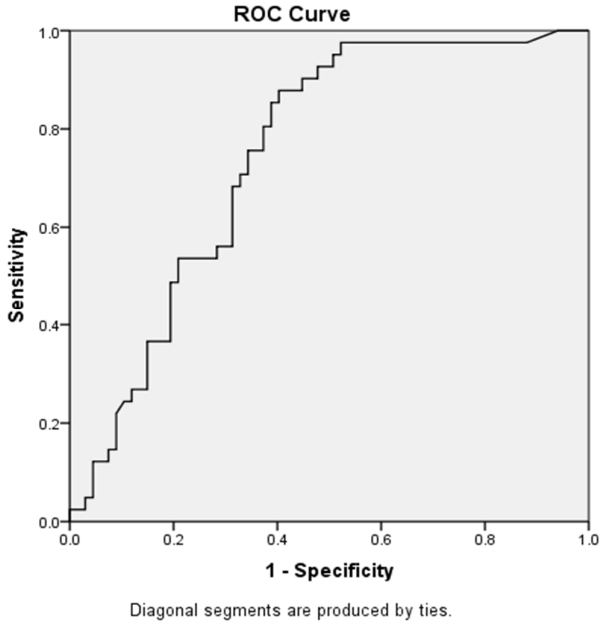
i) **Fig 5.20** Conjunctive rule



j) **Fig 5.21** K star



k) **Fig 5.22** nnge



5.5 Inter project-validation of prediction model using J-security

We cross-validated prediction models for change prone classes constructed using OO metrics evaluated from J-Meter software with the help of machine learning models. The cross-validation is done using test set from J-Security software. BayesNet model outperformed other machine learning models with maximum AUC value of 0.764 with 87.5 percent sensitivity and 58.1 percent specificity. Naïve Bayes and Random models given competitive results with AUC values 0.736 and 0.715.As we can see in table 9 that AUC for j48 and adaboost is also quite similar.

Table 5.7 Inter project validation result

S.N	Method used	Area Under Curve	Sensitivity	Specificity	Cut-Off point
1	Bayes Net	.764	87.5	58.1	.397
2	Naïve Bayes	.736	71.9	.71	.1145
3	J48	.685	58.9	.621	.642
4	Random forest	.715	82.5	.611	.452
5	Adaboost	.673	53.1	.532	.558
6	Bagging	.632	53.6	.611	.456
7	Part	.561	72.0	.454	.762
8	Conjunctive rule	.575	53.5	.652	.657
9	K star	.615	57.3	.629	.509
10	nnge	.629	53.6	785	.563

5.6 Discussion of results

Validation result of our study on j-Meter and XML security suggest that random forest and adaboost are the best methods as compared to others for prediction of change prone classes .Methods like Naïve bayes and Bays net have given comparable results in both the software's as well as in cross validation model. Another important thing that we have gone through is that metrics like LCOM, CBO, RFC are good indicator of change proneness. Inter project validation models is also giving good results showing that already existing models can also be used for the prediction of change prone classes. These prediction models can be used in the early phase of software development so that we can give some more resources for the testing of change prone classes since these will require more effort in the maintenance phase. Thus early prediction of these change pone classes can save our efforts in testing and maintenance and we can do better planning of our resource allocation.

Chapter 6

CONCLUSION AND FUTURE WORK

It has been seen from various studies that the change proneness plays a vital role not only in maintain the quality of the software but also it have also effects the maintenance of the software directly or indirectly. Since in this era of change in technology and functionality it is quite difficult to maintain the quality and maintenance is also taking a lot of cost so it is becoming very importance to analyze the relationship among the object oriented metrics and change proneness keeping this thing in mind we have done some analysis an open source software J-Metre that contains 336 classes metric WMC and RFC plays a vital role in the prediction now we have found some metrics that have good impact on the prediction of fault proneness .Now we have applied 10 machine learning techniques that checks the accuracy of our models by analyzing sensitivity , specificity and area under the curve of ROC. We have also found cut off points that shows a balance between the change prone classes and the classes that are not change prone. We have also seen that Adaboost and Bayes Net are giving the best result and hence the researchers working in this field can use these two techniques for the analysis of the data set. Hence it is advisable to use the model in early phases of the software developments to predict the quality of the software and prevent the software from so many hazards that can come in the maintenance phase.

Our main results are summarized as follows:

1. The metric LCOM and CBO is a significant indicator of change proneness as in both the three data sets, these two metrics was a representative of change proneness prediction, after applying feature subset selection method CFS.
2. Random Forest and Adaboost methods outperformed the LR model although all methods yielded good AUC using ROC analysis. We conclude that machine learning methods are comparable and competitive to LR. This study confirms that construction of models using machine learning is feasible, adaptable to OO systems and useful in predicting change prone classes. While research continues, practitioners and researchers may apply machine learning methods for constructing the model to predict change prone classes.
3. The predicted results can be used by researchers and practitioners in early phase of software development in order to reduce maintenance effort and costs. Thus the predicted models can be used in effectively and efficiently planning of testing resources

In future work we will consider some more factors like current environment, size of the software that can affect the change proneness. Some similar type of analysis can be done big data seta to get more results that can really help in the development of the quality products. Also in our next study we would like to concentrate on the some optimization of effort in the maintenance phase by change proneness.

REFERENCES

1-Yauv freund and Robert E schapire "*A decision theocratic generalization of online learning, and an application to boosting*". Journal of computer and system science 2004.

2- KK Aggarwal, Y Singh, A Kaur, R Malhotra "*Empirical study of object-oriented metrics*". Journalo9f Object Technology 5(8):149–173 2006.

3- RuchikaMalhotra ,Megha Khanna" *Investigation of relationship between object-oriented metrics and change proneness*" Springer-Verlag April 2012.

4- F. Brito e Abreu , W. Melo, "*Evaluating the Impact of Object-Oriented Design on Software Quality*," *Proceedings Third Int'l Software Metrics Symposium*, 1996, pp.90-99

5- J. Bansiya and C. Davis, "*A Hierarchical Model for Object-Oriented Design Quality Assessment*," *IEEE Trans. Software Eng.*, Vol.28, No.1, 2002, pp.4-17.

6-K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "A *validation of object-oriented metrics*," *NRC Technical report ERB-1063*,1999.

7-Ruchika Malhotra , Ankita jain "*Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality*" Journal of Information Processing Systems, Vol.8, No.2, June 2012.

8-http://jmeter.apache.org/

9- S. Chidamber and C. Kemerer, "*A Metrics Suite for Object-Oriented Design*," *IEEE Trans. Soft Ware Eng.*, Vol.20, No.6, 1994, pp.476-493.

10- M Lorenz, J Kidd *"Object-oriented software metrics"*. In: Prentice Hall object-oriented series. Prentice Hall, Englewood Cliffs 1994.

11- Sharafat AR, Tavildari L Change prediction in object oriented software systems: a probabilistic approach in 11th European conference on software maintenance and reengineering june 2007.

12-*M.H. Tang, M.H. Kao, and M.H. Chen, "An empirical study on object-oriented metrics," In Proceedings of Metrics, 242-249.*

13- Zhou Y, Leung H, Xu B Examining the potentially confounding effect of class size on the associations between object oriented metrics and change proneness. IEEE Trans Softw Eng 35(5):607–623 may 2009.

14- MA Chaumum, H Kabaili, "*A change impact model for changeability assessment in object oriented software systems*" . Third European conference on software maintenance and reengineering, pp 130 1994.

15- Tsantalis N, Chatzigeorgiou A, Stephanides G (2005) Predicting the probability of change in object oriented systems. IEEE Trans Software Eng 31(7):601–614.

16-Foutse Khomh ,Massimiliano Di Penta, Yann-GaëlGuéhéneuc "*An exploratory study of the impact of antipatterns on class change- and fault-proneness*" Springer Science+Business Media, LLC 2011.

17-AR Han , S Jeon , D Bae , J Hong "*Behavioural dependency measurement for change proneness prediction in UML 2.0 design models*" computer software and applications 32nd annual IEEE international 2008.

18- D'Ambros M, Lanza M, Robbes R "*The relationship between change coupling and software defects*" 16th working conference on reverse engineering, pp 135–144 2009.

19- Bieman J, Straw G, Wang H, Munger PW, Alexander RT (2003) Design patterns and change proneness: an examination of five evolving systems. In: The proceeding of 9th international software metrics symposium.

20- http://www.statsoft.com/textbook/naive-bayes-classifier/

21- http://www.bayesnets.com/

22- http://homes.cs.washington.edu/~pedrod/papers/mlc04.pdf

23- http://en.wikipedia.org/wiki/C4.5_algorithm

24-http://www.dabi.temple.edu/~hbling/8590.002/Montillo_RandomForests_4-2-2009.pdf

25-L. Breiman, "*Bagging Predictors*". Machine Learning journals, vol.24, no. 2, pp. 123-140, 1996.

26- Braga and Adriano L. I. Oliveira "*Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals*" 19th IEEE International Conference on Tools with Artificial Intelligence.

27- *"Multiple Classifiers Applied to Multisource Remote Sensing Data"* IEEE transactions' on geosciences and remote sensing, vol 40, no. 10, october 2002.

28-Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines by john c platte (http://www.bradblock.com/Sequential_Minimal_Optimization_A_Fast_Algorithm_for_Training_Support_Vector_Machine.pdf)

29- MohdFauzi bin Othman ,ThomasMoh Shan Yau *"Comparison of Different Classification Techniques Using WEKA for Breast Cancer "* Control and Instrumentation Department, Faculty of Electrical Engineering, UniversitiTeknologi Malaysia, Skudai, Malaysia

30- Leonard E. Trigg K: *"An Instance-based Learner Using an Entropic Distance Measure"*. Department of Computer Science, University of Waikato, New Zealand.

31-K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "A *validation of object-oriented metrics,*" *NRC Technicalreport ERB-1063*,1999.

32-Y. Singh, A. Kaur, and R. Malhotra, "*Empirical validation of object-oriented metrics for predicting faul tproneness models,*" *Software Quality Journal*, Vol.18,No.1, 2010,pp.3-35.

33-Understand Your Code – *List of Object Oriented Metrics Available* [online], http://www.scitools.com/documents/metricsList.php?metricGroup=oo .

34- M.Stone, "*Cross-valedictory choice and assessment of statistical predictions,*" Journal Royal Stat. Soc., Vol.36, 1974, pp.111-147.

35- M.English, C.Exton, I.Rigon and B.Clearyp, "*Fault Detection and Prediction in an open source Software project*," Proceedings of the 5th International conference on Predictor Models in Software Engineering 2009.