EVALUATION OF RELATIONSHIP BETWEEN OBJECT ORIENTED METRICS AND CHANGE PRONENESS

A Dissertation Submitted in partial fulfilment for the award of degree of

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

By

SAPANA KHARWAR

(Roll No. 2K12/SWE/24)

Under the guidance of

DR. RUCHIKA MALHOTRA

Department of Software Engineering

Delhi Technological University, Delhi

Department of Computer Engineering

Delhi Technological University, Delhi

2012-2014

DELHI TECHNOLOGICAL UNIVERSITY

CERTIFICATE

This is to certify that the project report entitled **EVALUATION OF RELATIONSHIP BETWEEN OBJECT ORIENTED METRICS AND CHANGE PRONENESS** is a bonafide record of work carried out by Sapana Kharwar (2K12/SWE/24) under my guidance and supervision, during the academic session 2012-2014 in partial fulfilment of the requirement for the degree of Master of Technology in Software Engineering from Delhi Technological University, Delhi.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Dr. Ruchika Malhotra

Asst. Professor

Department of Software Engineering

Delhi Technological University

Delhi

DELHI TECHNOLOGICAL UNIVERSITY

ACKNOWLEDGEMENT

With due regards, I hereby take this opportunity to acknowledge a lot of people who have supported me with their words and deeds in completion of my research work as part of this course of Master of Technology in Software Engineering.

To start with I would like to thank the almighty for being with me in each and every step of my life. Next, I thank my parents and family for their encouragement and persistent support.

I would like to express my deepest sense of gratitude and indebtedness to my guide and motivator, **Dr. Ruchika Malhotra**, Assistant Professor, Department of Software Engineering, Delhi Technological University for her valuable guidance and support in all the phases from conceptualization to final completion of the project.

I wish to convey my sincere gratitude to **Prof. Rajeev Kapoor,** Head of Department, and all the faculties and PhD. Scholars of Computer Engineering Department, Delhi Technological University who have enlightened me during my project.

I humbly extend my grateful appreciation to my friends whose moral support made this project possible.

Last but not the least, I would like to thank all the people directly and indirectly involved in successfully completion of this project.

Sapana Kharwar

Roll No. 2K12/SWE/24

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Changes are always inevitable. As the software evolves changes might occur due to some defect or when some additional functionality is added to the software. Change proneness is the probability of changing some part of software. The requirement modification is needed if some changes occur. If there are more changes needed in the software, then this means that there is a problem of design quality and therefore it's design needs to be improved. In such cases, it is very important to discover change prone classes in the software in early phases of software development so that testing resources can be planned to reduce the maintenance effort. As a result testing becomes more qualitative because more focus will be laid on those classes that are more prone to changes. By doing so, the probability of occurrence of defects can be reduced which can thereby lead to better maintenance. Our study analyzes the relationship between object oriented metrics and change proneness. Statistical and machine learning methods have been studied for predicting change prone classes. These methods have been applied on five open source java projects namely ABRA, ABBOT, APOLLO, AVISYNS and JMETER. The performance has been analyzed on the basis of receiver operating characteristics. Results have shown that the performance of machine learning techniques is comparable to statistical methods.

CHAPTER 1

INTRODUCTION

As the software evolves, lot of changes occurs in the software. There can be a lot of reasons that can lead to changes in software. Some of the reasons can be defects, adaptive or perfective maintenance etc where defect can be described as lack of some features that leads to accomplish the desired goal. Adaptive maintenance can be defined as change in the software programs after delivering it to the customer because of changing some requirements or if some functionality is not performing well and predictive maintenance is a technique that is designed to determine the current condition of software which is in service and prediction of when maintenance is required. So these days management of software system should be given more priority than increasing its completion goal. However there is more competition in software world that is why one should try to make software more correct in all aspects like outcomes, requirements, design etc so that there would be less maintenance cost or effort after delivering the software to its customer.

Software maintainability is most important aspect for all the organisations that are involved in development of large software. From the past decades it can be seen that maintenance of software includes 40-60% cost of the overall cost in development of software. Because of this reason maintainability of software is believed to be more challengeable problem in software organisations. So we should do something such that maintenance cost would be less. Maintainability of software means how we should modify the software code to remove the fault and improve the software's performance. We can also cop up with the changed environment. There are many factors that can lead to change the software product and these are: correction of identified faults, accommodate changed environment, to improve the performance of software or to increase the software source code understandability.

Extensive research has been done to show the relationship between various object oriented metrics and fault proneness of a software. These studies help in efficient utilization of resources. Here object oriented metrics are the basic unit to determine the characteristics of software. Fault proneness is defined as the probability of being fault in software. The question arises why change proneness prediction is needed? The answer is if we do not predict the change prone classes then maintenance cost will be high and customer satisfaction level will be less. So it will be always beneficial for developer to have some technique or some model so that it can be applied to detect software change prone classes. Here object

oriented metrics can be used to predict change prone classes. We need to examine the software metrics from the design phase as time and cost that is incorporated in maintenance of software is high. Because of these software metrics it will be easy for software developer to improve the various aspects like design of software, coding and implementation of software otherwise this could lead increase in maintenance cost in later phases of software development life cycle. Change proneness means the probability of changing certain part of software is high and it requires assessment. It can also be defined as probability of change that can occur. For software some parts may be more change prone than other part of software. Change proneness prediction may be very helpful since it demonstrate the software's design quality. If very high changes is needed in the modification of the software that means there is a design quality problem and it design needs to be improved. Some time there may be a probability to redesign the model also and again evaluate it. Prediction of classes that are prone to changes can help in maintenance and testing activities of software development life cycle. Maintenance consumes around 40-80% of the effort to develop a particular software. Testing of a change-prone class should be done very carefully when the software changes or some additional functionality is added to the software.

Our study analyzes the relationship between object oriented metrics and change proneness of a class. The performance of various machine learning algorithms and logistic regression to predict change prone classes have also been analyzed. For the purpose of validation we have taken five open source software namely ABRA, ABBOT, APOLLO, AVISYNS and JMETER. All these software are written in java language. Two versions of each software have been taken. Changes have been analyzed in term of number of lines added, deleted and modified in the new version as compared to previous version. The object oriented metrics for all the software are generated by using a tool named understand for java. We also generated change statistics by using a tool named CMS tool [16]. We merged these two files i.e. metric and change statistics file to yield the data points. The main objective of this study is to predict the change prone classes which would result in effective planning of testing activity. Here we will predict the best model for prediction of change proneness.

1.1 Motivation of Work

As we have seen above that, now-a-days maintenance is a challenging job. Software maintainability means how easily software can be modified so that an error can be removed or some extra functionality can be incorporated. From the past it can be seen that maintenance consumes 40-60% cost of the entire cost involve in software development. So we can say that software maintenance needs to be more taken care of as compared to other phases of software development and it is also challenging job. There is one more reason because of which maintenance is important and that is maintenance requires 40-80% effort to develop software. Sometimes because of a very little risk or fault software needs very high maintenance cost which causes customer dissatisfaction. So maintainability is needed as to remove the fault, add some new functionality etc. The reason of changing the requirement is adapt new environment, adapt new technology, performance improvement etc. Testing of a change-prone class should be done very carefully when the software changes or some additional functionality is added to the software.

For the above reason, we have done our study. So that we can find the change proneness prediction model so that it can be used to reduce the maintenance cost and effort both. In this study we analyses the relationship between object oriented metrics and change proneness of a class. The performance of various machine learning algorithms and logistic regression to predict change prone classes have also been analyzed.

1.2 Aim of Work

As it can be seen that these days two factors: maintenance and cost aspects and has to be examined more than other activities so that cost and effort should be reduced and also software can be delivered on time. Sometimes because of a very little risk or fault software needs very high maintenance cost which causes customer dissatisfaction. So to get rid of this problem we need to apply best method to predict the change proneness so that if any changes required in future, it can be done easily without taking more time and maintenance cost.

So the main purpose of our study is an empirical validation. Here we examine the relationship in change proneness of a class and object oriented metrics and analyze the performance of various machine learning methods and logistic regression to predict the change proneness classes. We have taken five open source software for the purpose of validation and the software names are: ABRA, ABBOT, APOLLO, AVISYNS and JMETER and all these are written in java language. Each software can have many versions but we have taken only two

version of each software. Changes have been analyzed in term of number of lines added, deleted and modified in the new version as compared to previous version. In this study we used a tool named ‘understand for java’ to object oriented metrics for each software. And we also used CMS tool to generate change report or statistics. And finally we merge the above two files to get the complete dataset. The objective of this study is to analyse the performance of various machine learning methods and logistic regression. If we successfully predict the change proneness then this result can helps us to do the effective planning of testing activity. Here we will predict the best model for prediction of change proneness.

1.3 Organisation of Thesis

This paper is organized as follows. Section 2 summarizes the related work. Section 3 summarizes research background in which we first describe our independent and dependent variables thereby empirical data collection process and then descriptive statistics of every software has been described with the help of tables. Section 4 describes research methodology in which we first describe all the machine learning methods and statistical method (logistic regression). In section 5 we explain our result analysis which we have done in our study. This chapter 5 includes univariate LR results and model evaluation using ROC curve. The last is Section 6, which describes conclusion and future work of our study.

CHAPTER 2

LITERATURE SURVEY

Han et al. [1] worked on improving the quality of design by predicting change proneness in UML 2.0 models. Here a method is used for rating classes according to probability of change. The method is named as BDM(Behavioural Dependency Measure).The evaluation of results is done on JFreechart which is a multiversion medium sized open source project. The final result of this work indicates that Behavioural Dependency Measure is an effective measure for prediction of change proneness.

Ambros et al. [2] used correlation and regression analysis in order to study the relationship between change coupling and software defects .Change coupling means the degree of dependency of the artifacts on each other because they emerge together. There is always a probability that a change that occur in one part of software might lead to changes in other part of the software. Inorder to study this three large software have been analyzed. Study shows that there exists a strong correlation between change coupling and defects that occur in software. Also change coupling has a strong correlation with severe defects as compared to minor defects.

Sharafat et al. [3] proposed a probabilistic model to predict the probability of a change in each class. This approach is based on probability calculates the change in a particular class based on its source code and change history. The source code is analyzed over different software releases using reverse engineering techniques in order to collect the code metrics. This proposed technique is analyzed on a medium sized system i.e JFlex which is a lexical analyzer for java codes.

Chaumum et al. [4] defined a change impact model to study the consequences of changes made to classes of the system. This model was defined in C++ language. There are mainly two factors on which the impact of change depends: (a) The type of change which can propagate to other classes and (b) The type of link that binds different classes. Two classes can be linked by association, aggregation or inheritance. The study in this case was done on telecommunication system.

Jhou et al. [5] shows that class size should be taken as a confounding variable while we are validating the impact of object oriented metrics on fault- proneness. For this purpose, they used three size metrics out of which two metrics are used in high level design. These two metrics are also used to determine the confounding effect of class size on association between object oriented metrics and change proneness of classes.

Design patterns are keys to common design complications [6]. Bieman et al. [6] examined five evolving systems to analyze the relation between design patterns and change proneness. The classes of four patterns out of five were less change prone. Only one design pattern was more change prone. Therefore, the study helps us in studying design patterns that are more adaptable.

Tsantalis et al. [7] Amongst all the benefits of object oriented paradigms ,flexibility is the most important as our requirements keep on changing. But it is very difficult to quantify flexibility. So this study defines a probabilistic approach to estimate the change proneness of an object-oriented design by evaluating the probability that each class of the system will be affected when new functionality is added or when existing functionality is modified. This study uses two multiversion open source projects to evaluate the proposed model. Here java is used to automate the whole process and statistical analysis is used to improve the correlation between extracted probability of each class and the actual changes in each classes.

In the paper by Lindvall [8], researcher monitors and measures the capability of experienced software developers to predict software change caused by new requirements to an existing software system (i.e. impact analysis) at different levels of granularity.

Malhotra et al. [9] investigated the relationship between object oriented metrics and change proneness. So change proneness classes of software can be identified using software prediction model which is based on the results obtained from this study. By using this model we will get better result. This paper uses statistical and machine learning methods to predict the software quality. They evaluate and compare the performance of these machine learning methods with statistical method (logistic regression). Thus, the developed models can be used to reduce the probability of defect occurrence and better maintenance can be achieved.

The paper by Ingram et al. [10] shows that software requirement and design can also be used to make prediction. They define a set of metrics to obtain data from software requirement and design phases and used a case study project to obtain the value of these metrics. Here it can be seen that there is a significant difference in change proneness between the components with high or low value of these metrics.

Romano et al. [11] find that anti-patterns are poor solutions to design and implementation problems and due to this reason object oriented system becomes difficult to maintain. So this paper focused on these problems by considering fine-grained source code changes (SSC) which are obtained from 16 java open source systems.

Malhotra et al. [12] used adaptive neuro-fuzzy inference system (ANFIS) to calculate the change proneness for the two commercial open source software systems. The performance of adaptive neuro- fuzzy inference system (ANFIS) can be compared with the other statistical and machine learning methods. These methods can be logistic regression, bagging, decision tree, multilayer perceptron, k-star etc. The effectiveness of the model can be determined by using receiver operating curve (ROC). According to this study ANFIS shows best result as compared to all other methods (statistical and machine learning methods).

Malhotra et al. [13] worked on improvement the software quality by using machine learning and statistical methods. They some metrics are used to predict a model to evaluate fault proneness. The metrics are object oriented CK metrics and QMOOD metrics. They used 6 machine learning methods and one statistical method. Area under curve (AUC) is used to analyse the result which is obtained from receiver operating curve (ROC). According to this paper the model predicted using random forest and bagging methods are best and they outperformed other methods.

Malhotra et al. [15] worked on reuse of generated prediction model of one project and validates it on another project. This study used two open source project for evaluation. Both the project is written in java language. Receiver operating curve (ROC) is used to examine the performance of predicted model. The final result of this work indicates that training sets are successfully applied for validation of inter project. Time and effort can be optimized using these results which are required to create training data for each project. The advantage of this work is that resources and time can be effectively utilised.

CHAPTER 3

RESEARCH BACKGROUND

3.1 Independent and Dependent Variable

In this section, we describe independent and dependent variable used in our study.

3.1.1 Independent Variable

In this study, Object Oriented metrics are used as independent variables. Since single metric is insufficient to discover all the characteristic of software under development so we have used fifteen metrics. In this study we have selected only those metrics which are most significant to exhibit the software characteristics like inheritance, cohesion, coupling etc. All the metrics and values of these metrics are obtained by using a tool named 'understand for java'. All the fifteen metrics i.e. independent variables are described in Table 3.1.

Table 3.1 Object Oriented Metrics

Serial No.	Metric name	Description
1	Coupling between object(CBO)	CBO is count of number of other classes to which a class is coupled.
2	Number of children(NOC)	NOC is defined as the number of immediate subclasses from a given class.
3	Number of class method (NOM)	It is defined as the total number methods in a given class.
4	Number of class variable (NOA)	It is defined as the total number of variables in a given class
5	Number of instance method(NIM)	It is the count of total number of instance method.
6	Number of instance variable(NIV)	It is defined as the total number of instance variable.
7	Number of local methods(NLM)	NLM is defined as the total number of local variable of a given class.
8	Response for a class(RFC)	It is the count of number of methods that can be executed when a message from an object of a given class is received.

9	Number of local private methods (NPRM)	It is defined as the total number of local private methods which are not inherited.
10	Number of local protected methods (NPROM)	It is the total number of local protected methods which are not inherited.
11	Number of local public methods (NPM)	It is the total number of local public methods which are not inherited.
12	Lines of code (LOC)	Total number of lines of code in a given class.
13	Depth of inheritance tree(DIT)	DIT is defined as the maximum length from a class node to the root of the inheritance tree.
14	Lack of cohesion in methods (LCOM)	Total count of pair wise local methods in a class having no variable or attribute in common.
15	Weighted method per class (WMC)	WMC is defined as the total number of sum of cyclomatic complexity of all methods in a given class.

3.1.2 Dependent Variable

In software development, maintainability requires necessary modifications. So advance knowledge of change prone classes can help us to minimize extra maintainability cost of software. In this study, we have taken CHANGE as dependent variable. Change can be analysed as number of lines added, number of lines deleted or number of lines modified. In this paper, we examine the relationship between change proneness and object oriented metrics. To examine the change proneness we use receiver operating characteristic (ROC) analysis.

3.2 Empirical Data Collection

In this section, we describe data sources and give detailed description of each data sources and data collection method.

We analyzed five open source software which are written in java. We found all five software from 'sourceforge.net'. We downloaded two different version of each open source code used.

In this study, we analyze changes in every class of both versions of all five software. Software we studied are *ABRA, ABBOT, APOLLO, AVISYNC, JMETER* and are summarized

in table 3.2. Table 3.2 shows the programming language, version number, number of changed classes, number of unchanged classes, total number of common classes and percentage of changed classes.

Table 3.2: Description of each software

Dataset	**Programming language**	**Version 1**	**Version 2**	**Number of changed classes**	**Number of unchanged classes**	**Total Number of common classes**	**% of changed classes**
Abra	Java	0.9.8	0.9.9	33	147	180	18.3
Abbot	Java	1.0.0rc1	1.0.0rc3	86	241	327	26.3
Apollo	Java	0.1	0.2	69	183	252	27.4
Avisync	Java	1.1	1.2	27	46	73	36.9
Jmeter	Java	2.8	2.9	537	363	900	59.7

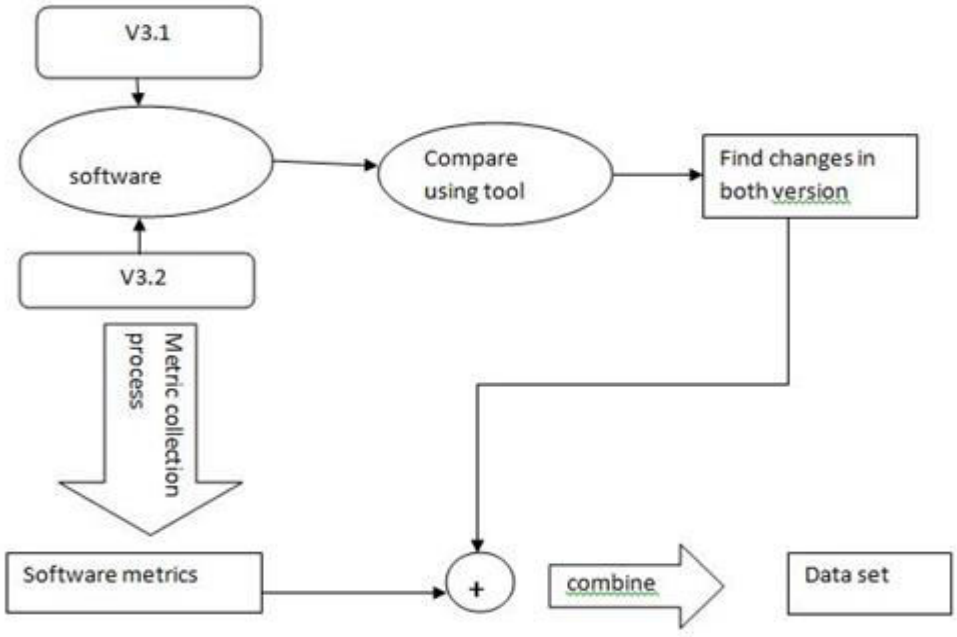


Fig 3.1 Data Collection Process

From the figure it can be seen that we have to take software with two different versions. For example we take ABOT software with version v1 and v2. After applying metric collection process for which we used understand tool and after that we get a file which contains all the metrics with their values. We also compare both the versions of software to get change statistics or change report and this work is done by using CMS tool. Then we combine both these files to get final dataset.

3.2.1 Data Collection Method

In this section we will explain how to collect data points for all five software. Since we have to examine all the changes in every class files of both versions (previous and current version) and values for all the metrics we have used in our study and then merge both the files (change report and metric file). To obtain complete dataset we follow some steps and these steps are:

Step1. Metric Generation Process:

As we know metric is the basic unit to examine the characteristics of software. In this process we downloaded source code of two different version of five open source software (one previous and one current) as shown in Table 2 from 'Sourceforge.net'. We generated metrics for the first version of all five software (ABRA-0.9.8, ABBOT- 1.0.0rc1, APOLLO-0.1, AVISYNC-1.1, JMETER-2.8) with the help of "understand tool" for java. The metric file obtained in this contains metric for all classes (because in this study we are analysing changes in the classes of both version of software), methods and unknown classes which we discard while we make dataset. At the end of this stage we now have metric file for all software we have used above.

Step2. Pre-processing Step:

Since we have to find only common classes so in this step, we perform data filter to extract common classes which are common in both the versions of each software (current and previous version).

Step3. Change Report Generation:

To generate change report we used CMS tool [16] for java. CMS tool works as follows: first we open CMS tool, then we click run button, it will display a popup window which demands both the versions of software [16]. Then we click on compare button and finally it will

generate change report of CSV extension. We repeat this process for other four software and we get change report of all five software.

Step4. Merging Files:

In this step, we combine metric file obtained in step 1 and change report file obtained in step 3 to yield complete dataset. Here we search common java classes in both the files and then merge them. At the end of this step we have complete dataset of all five software.

3.3 Descriptive Statistics

Here we will describe statistics results of each software. Tables 3.3, 3.4, 3.5, 3.6 and 3.7 shown below are the descriptive results that contain mean, median, mode, standard deviation, variance, minimum, maximum and percentiles for each metrics of all software used. From tables 3.3, 3.4, 3.5, 3.6 and 3.7, we can see that the mean value of number of children (NOC) for software (ABRA-0.73, ABBOT-0.73, APOLLO-0.69, AVISYNC-0.56, JMETER-0.52) which is very low for each software, so we can conclude that number of children are very less i.e. inheritance is not much used in all five systems. LCOM metric which is defined as the total count of classes having no attribute or variable in common has greater value in all the systems (approximately 100). Similar results have been shown by other researchers [17, 18, 19].

Table 3.3 Descriptive Statistics Result for ABRA Dataset

	Mean	Median	Mode	Std. Deviation	Variance	Minimum	Maximum	Percentiles 25	Percentiles 75
CBO	3.27	2.00	0	4.286	18.367	0	24	1.00	4.00
NOC	.73	0.00	0	2.234	4.990	0	13	0.00	0.00
NOM	.90	0.00	0	1.658	2.750	0	7	0.00	1.00
NOA	1.03	0.00	0	2.574	6.625	0	23	0.00	1.00
NIM	7.99	5.00	0	11.161	124.559	0	67	2.00	9.00
NIV	3.04	2.00	0	5.065	25.658	0	36	0.00	4.00
NLM	8.89	6.00	3	10.745	115.451	0	67	3.00	9.00
RFC	18.71	10.00	6	23.461	550.430	0	95	6.00	17.00
NPRM	.60	0.00	0	2.315	5.359	0	14	0.00	0.00
NPROM	2.19	0.00	0	6.626	43.901	0	51	0.00	1.00
NPM	5.82	4.00	4	6.471	41.871	0	64	3.00	7.00
LOC	81.44	35.50	16	158.811	25220.941	2	932	18.00	63.75
DIT	1.81	2.00	1	.908	.824	1	4	1.00	2.00
LCOM	49.32	60.00	0	35.421	1254.644	0	100	0.00	80.00
WMC	17.77	8.00	5	31.917	1018.694	0	165	5.00	14.75

Table 3.4 Descriptive Statistics Result for ABBOT Dataset

	Mean	Median	Mode	Std. Deviation	Variance	Minimum	Maximum	Percentiles	
								25	75
CBO	23.06	5.00	113	40.384	1630.893	0	113	2.00	13.00
NOC	.73	0.00	0	2.867	8.222	0	44	0.00	1.00
NOM	1.98	0.00	0	5.829	33.972	0	55	0.00	3.00
NOA	3.89	1.00	0	5.999	35.985	0	28	0.00	5.00
NIM	27.43	8.00	112	39.814	1585.173	0	112	3.00	29.00
NIV	11.51	2.00	0	18.765	352.122	0	52	0.00	10.00
NLM	29.42	9.00	115	40.999	1680.907	0	115	4.00	29.00
RFC	60.05	30.00	115	64.276	4131.430	0	221	8.00	115.00
NPRM	15.59	1.00	0	31.077	965.788	0	85	0.00	6.00
NPROM	1.72	1.00	0	3.583	12.835	0	32	0.00	2.00
NLM	10.34	6.00	19	12.162	147.906	0	86	2.00	19.00
LOC	574.21	106.00	2656	949.358	########	2	2656	39.00	485.00
DIT	2.24	2.00	1	1.387	1.923	1	6	1.00	3.00
LCOM	59.32	72.00	0	37.294	1390.825	0	98	19.00	93.00
WMC	86.20	19.00	369	133.556	17837.243	0	369	7.00	72.00

Table 3.5 Descriptive Statistics Result for APOLLO Dataset

	Mean	Median	Mode	Std. Deviation	Variance	Minimum	Maximum	Percentiles	
								25	75
CBO	5.74	5.00	0	5.242	27.475	0	35	2.00	9.00
NOC	.69	0.00	0	3.630	13.177	0	44	0.00	0.00
NOM	.71	0.00	0	2.025	4.101	0	11	0.00	0.00
NOA	1.33	0.00	0	2.826	7.984	0	14	0.00	1.00
NIM	8.29	5.00	5	8.804	77.512	0	82	4.00	11.00
NIV	3.69	2.00	0	4.201	17.649	0	30	1.00	6.00
NLM	9.00	5.00	5	9.131	83.371	0	82	4.00	11.00
RFC	15.23	13.00	5	11.594	134.417	0	83	5.00	23.75
NPRM	.39	0.00	0	1.018	1.035	0	7	0.00	0.00
NPROM	.42	0.00	0	1.028	1.057	0	9	0.00	0.00
NPM	7.94	5.00	5	8.213	67.451	0	74	3.00	10.00
LOC	110.82	49.00	17[a]	144.418	20856.508	2	1024	26.00	130.00
DIT	1.85	2.00	2	.737	.543	1	4	1.00	2.00
LCOM	47.44	50.00	0	33.241	1104.949	0	100	13.00	77.75
WMC	18.73	12.00	5	22.862	522.678	0	229	5.00	24.00

Table 3.6 Descriptive Statistics Result for AVISYNC Dataset

	Mean	Median	Mode	Std. Deviation	Variance	Minimum	Maximum	Percentiles	
								25	75
CBO	3.74	1.00	0	5.273	27.806	0	27	0.00	8.00
NOC	.56	0.00	0	1.472	2.166	0	9	0.00	0.00
NOM	.22	0.00	0	1.170	1.368	0	9	0.00	0.00
NOA	2.27	1.00	0	3.568	12.729	0	17	0.00	2.00
NIM	8.05	6.00	1	7.612	57.941	0	32	1.00	11.00
NIV	2.08	2.00	0	2.707	7.326	0	14	0.00	3.00
NLM	8.27	7.00	1	7.653	58.563	0	32	1.00	11.00
RFC	15.08	9.00	5[a]	12.246	149.965	0	44	6.00	24.00
NPRM	1.85	0.00	0	4.300	18.491	0	23	0.00	2.00
NPROM	.07	0.00	0	.585	.342	0	5	0.00	0.00
NLM	6.36	6.00	1	6.005	36.066	0	32	1.00	8.50
LOC	61.23	36.00	5	71.205	5070.209	4	359	6.50	87.50
DIT	2.26	2.00	1	1.334	1.779	1	5	1.00	3.00
LCOM	70.73	85.00	100	34.751	1207.646	0	100	59.00	98.00
WMC	12.05	9.00	1	13.197	174.164	0	68	1.50	15.00

Table 3.7 Descriptive Statistics Result for JMETER Dataset

	Mean	Median	Mode	Std. Deviation	Variance	Minimum	Maximum	Percentiles	
								25	75
CBO	5.37	4.00	1	6.072	36.874	0	41	1.00	7.00
NOC	.52	0.00	0	2.710	7.347	0	51	0.00	0.00
NOM	1.10	0.00	0	4.593	21.096	0	67	0.00	0.00
NOA	3.94	2.00	1	7.539	56.844	0	75	1.00	4.00
NIM	10.02	6.00	1	12.856	165.267	0	111	3.00	13.00
NIV	3.25	1.00	0	5.928	35.143	0	55	0.00	4.00
NLM	11.11	7.00	5	13.446	180.800	0	116	4.00	15.00
RFC	30.56	16.00	3	32.982	1087.840	0	206	6.00	43.00
NPRM	1.84	1.00	0	3.175	10.082	0	20	0.00	2.00
NPROM	.58	0.00	0	1.958	3.832	0	32	0.00	0.00
NPM	8.53	5.00	2	11.553	133.477	0	102	3.00	10.00
LOC	114.20	65.00	5	145.005	21026.482	2	1095	28.00	134.75
DIT	2.27	2.00	2	1.199	1.437	1	5	1.00	3.00
LCOM	67.00	76.00	0	29.875	892.494	0	100	58.00	88.00
WMC	21.45	12.00	5	28.657	821.238	0	230	5.00	26.00

CHAPTER 4

RESEARCH METHODOLOGY

4.1 METHODS

4.1.1 NAIVE BAYES (NB)

It is a probability based classifier and makes use of bayes theorem of probability. Here supervised learning technique is used and they can be trained very efficiently. One of the greatest advantages of using naive Bayesian classifier is that they require very small amount of training data. Naive bayes classifier reduces the complexity by assumption of conditional independency.

Conditional independency is defined as follows:

Suppose we are given three variables randomly x, y and z. We can say that x is conditionally independent of y given z if and only if the probability distribution determining x is independent of value of y given z [23].

Naive Bayes classifier is a combination of parameters and structure. It has a star like structure and this structure need not to learn. So the core objective of NB classifier is to estimate the parameters using complete data so that it can be used to learn the classifier. The structure of this classifier is shown below in figure 4.1.1.

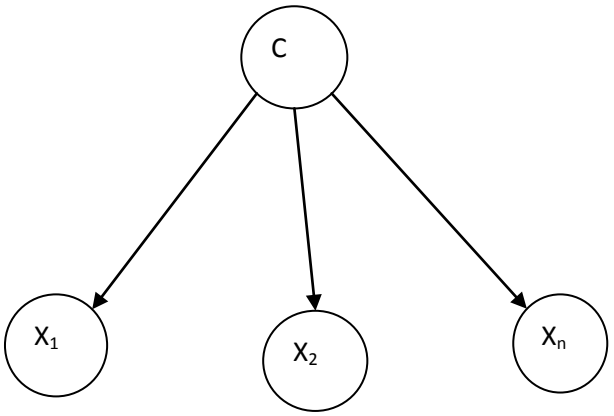


Fig 4.1.1 Naive Bayes Architecture

In the above figure x_1, x_2,.....x_n are attribute variables which are independent of each other for a given class variable C.

$$P(c_k) = \widetilde{P(c_k)} = (N_{ck})/N$$

Where,

N_{ck} is the number of evidence of ck^{th} class.

Advantages of Naive Bayes

- It is very fast and space efficient method used for training (single scan) and also fast to classification. We can take the example of database in which we can look for the probability with single scan in the database and it can be stored into a table, this storing process can taken as classification.
- Irrelevant features have no impact on this method.
- It can handle real as well as discrete data.
- Streaming data can be handled by it efficiently.

4.1.2 MULTILAYER PERCEPTRON (MLP)

Multilayer perceptron is the most famous neural network. We can use this model estimation/ computation. Multilayer perceptron is used for finding the patterns into data by analysing relationship between input to the neurons and their corresponding outputs. The relationship can be simple or complex.

It is a feed-forward artificial neural network model where there is mapping done from set of particular inputs to outputs. Single output is computed from inputs by forming a linear equation in accordance with the weight of links and then passing that linear equation through non-linear activation function. It contains multiple layers (more than one). Each layer contains multiple nodes which are fully connected to the next layer to it in a directed graph. Each layer receives input from its immediate preceding layer and provides output to just next succeeding layer to it. The overall result of network is the total output from each neuron in the output layer of network. MLP uses supervised learning method named Back Propagation which is very general learning paradigm to train the network. It comprises two passes, forward pass and backward pass. In the forward pass a set of input is supplied to the network and a set of output is produced as actual result from the network. Here synaptic weight is used which is kept fixed and the impact of this weight is propagated though every layer in the network. And in backward pass, it prompt/indicate an error which is the difference between the desired and actual response of the feed forward network. This error signal is propagated back so that we can readjust the weight to make desired response closer to the actual

response. MLP is an extension of linear perceptron. Non-linearly separable data can be distinguished by using this method.

Activation Function for MLP:

In the case where MLP uses linear activation function where this function maps input to output of neuron, we can reduce a network with more than two layers into two layer model with the help of linear algebra theory. It uses two activation functions:

$$\phi(v_i) = \tanh(v_i) \quad \text{and} \quad \phi(v_i) = (1 + e^{-v_i})^{-1}$$

Where,

The first activation function is known as hyperbolic tangent having range -1 to 1 and the second activation function is known as logistic function with range 0 to 1.

y_i is output of ith neuron.

v_i is weighted sum of neurons.

Layers of Multilayer Perceptron

The fig.4.1.4 shows the organisation of multilayer perceptron layers. The figure shows that there are three layers:

1. Input layer (left one) with three neurons
2. Hidden layer (middle one) which contains three neurons
3. Output layer (the right one) with three neurons

These three layers are explained as follows:

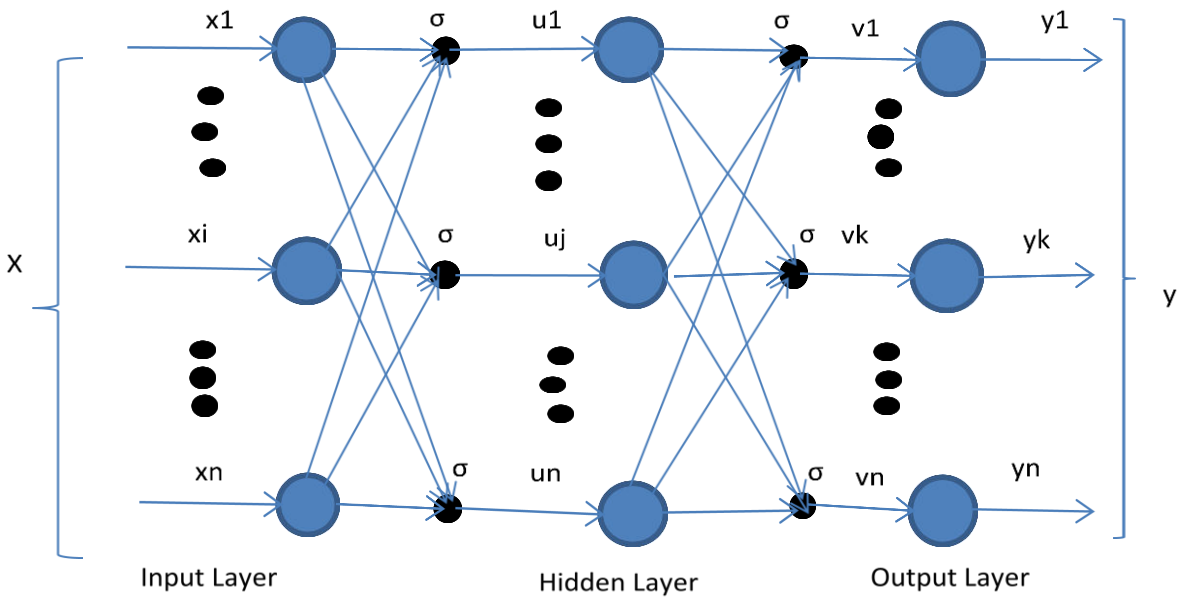


Fig 4.1.2 Multilayer Perceptron Architecture

Input Layer

A set of predictor variable (values x_1, x_2, x_3,... x_i,.....,x_p) are given as an input to the input layer. These values are processed so that these can be standardised to make the range of variables from -1 to 1. The values of each neuron is distributed in the hidden layer by the input layer. Here one more constant named bias other than predictor variable is used with value 1.0 given to the hidden layer. The value of bias is multiplied by weight factor and total sum (u_j) is fed into the hidden layer.

Hidden Layer

Hidden layer receives input which is the output of input layer. The input to each neuron in the hidden layer is multiplied by weight factor. These multiplication are added corresponding to each input which produces output as u_j . This u_j is fed into a transfer function σ which produce an output h_j . The output of hidden layer is given to the output layer.

Output Layer

In output layer, the input received from hidden layer is multiplied by weight factor wkj and the resulting values are added to produce a combined value vk this vk value is fed to the transfer function σ and we get yk as an output. These values of y are the output of the multilayer feed forward network.

Applications of MLP:

Regression analysis: If we perform regression analysis with continuous variables, then output layer contains single neuron and single y value is generated by the network.

Classification: If we perform Classification with categorical variables, then output layer contains N neuron and in this case network produces N value (one for each category) as an output.

Advantages of MLP

- Adaptivity
- Non-linearity
- Parallel architecture
- Fault tolerance.

4.1.3 K-STAR

K-Star is basically an instance based learning algorithm. The underlying assumption is that similar instances will have similar class. K-star can be considered as a class which contains test instances. Here K-nearest neighbour algorithm is used to calculate the degree of similarity. There is a difference between K-Star and other instance-based learning methods and the difference is K-Star uses a function named entropy based distance function and others does not do this. K-Star uses the concept of entropy for distance measure. To work with symbolic attribute K-Star uses the concept of entropy distance measure in ad-hoc/ unstructured manner. When classifier deals with missing data then this type of problem occurs. We can handle this problem either by treating each of the missing values as separate value and handle them differently means there should be maximum difference in each values as much as possible and replace values with average value, or simply ignore the missing values. For distance measure we use entropy, here distance will be computer between two instances. We can define the distance between two instances in the form of complexity of mapping one instance to another instance. The complexity can be calculated via two steps:

1. A finite number of transformations transform instances to instances.
2. We use a program that maps one instance to another instance. This program is a sequence of mapping that starts at A and terminates at B.

4.1.4 BAGGING

Bagging is also known as bootstrap aggregating. It creates various versions of training set to improvise classification models. It is a technique that gets data (using replacement method) from learning dataset by using a method named uniform probability distribution. Since we get data by using replacement method, it may be possible that some of the instances may present many times in the same training dataset and some of the others may be omitted from the dataset. It is a machine learning meta-algorithm which is used to improve the accuracy and precision of machine learning algorithms. It uses the concept of reducing variance and hence overfitting is avoided to a great extent. Usually it is used with decision tree methods. The main idea is to create different similar type of training sets and train develop new function for each of the training data. In case of class prediction majority of result will be selected. In classification, bagging uses majority vote on classification result while in regression it considers average of predicted values. For creation of multiple versions, we can go through the following steps:

1) Create bootstrap duplicates of learning set.
2) Use these learning sets as a new learning set.

Success behind bagging is its changeability of prediction method. If we do some changes in learning sets and it is causing critical changes in predictor developed, in such case bagging gives good results.

Suppose we have a training set Tn which contains N instances from population P. We also have a bootstrap set BSi which would contain N instances. Size of each bootstrap set is equal to the size of original data. For each instances of BSi, we can elicit any examples/ instances from the training set T individually and with repetition. As a result we may see that some instances are repeated in BSi and some examples are present in T but they are not present in BSi. So on average, we can say that bootstrap set BSi can include 63% data of original training dataset because each of the instances has the probability of 1-(1-1/N)N so that it can be selected in each of the BSi. If N is large enough, the above probability converges to 1-1/e= 0.632. Thus we can say that BSi is more likely as Tn but it is derived from T instead of P. We can combined all the bootstrap learning sets by taking average of all the outputs and in this way we can get an aggregated predictor [20, 21]. After training the k classifiers, a test instance is assigned to the class that receives the highest number of votes [22].

Bagging algorithm can be explained as follows:

- We are given training set with N instances and a class of learning methods for example, neural network, decision tree, naive bayes etc.
- We train multiple methods (more than one method) on various samples (obtained by data spliting) and calculate the average of predictions results of selected trained models (for example k models).
- The goal is to improve accuracy of one method with the help of its various copies and it gives quite better result in prediction capability of learning method when we take average of misclassification.

Advantages of Bagging Method

1) Bagging can provide significant gain to determine the accuracy of classification in comparison to other regression models.
2) It reduces variance.
3) Overfitting can be avoided by using bagging.

4.1.5 RANDOM FOREST (RF)

Random forest is a type of decision tree classification algorithm and it is developed by Breiman. Random forests are the methods used for learning and these methods work by constructing a series of decision trees at the time of training. Each tree of the random forest is made up of training sets which are selected randomly by using replacement algorithm. Random forest is a collection of tree predictor in which each tree of forest is relies on the value of random vector. The random vectors are handled independently for every tree in the forest. And also these vectors will be sampled with same distribution. So we can say that random forest is as a classifier which contains many decision trees. Finally, the final output class is taken as the mode of the output classes of each individual tree. To build a random forest, we initially mention how many decision tree we want to have in the forest and each tree should be fully grown means it should be of its maximum size.

In random forest, tree can be constructed using the following algorithm:

Suppose we are given N learning/ training samples and M variables in classifier. Here we declare a variable m such that m<<M. In the tree m demonstrate the total count of input (in the form of decision node) at the node of the tree. We also select a bootstrap learning sample. All the remaining training samples will be employed for estimation of error in the tree. This can be done by the prediction of their classes. The error rate of random forest depends on the two factors:

1) Correlation: correlation can be found between any of the two trees of forest. If correlation increases that means error rate also increases.
2) Strength: each individual tree of the random forest has its own strength. If a tree has low error rate will be considered as stronger classifier. While increasing the strength it will decrease the error rate.

If we decrease the value of m, it decreases the correlation and strength both. And both will be increases if we increase m.

To build the decision tree we use random subset of variables which are available. For each node in the tree m can be chosen randomly. All m variables which we have chosen help us to find the best split in training samples. These variables help us how data samples can be best partitioned at every node of the tree. We demonstrate the final result by majority. Each of the trees in the forest gives its own result in the form of vote and majority will be selected and that tree wins. By using above method we are able to get a tree which is fully grown and it is not pruned also.

Random forest algorithm can be understood stepwise as follows:

Let N is the number of trees which we want to build. For each of N trees, do the folloing iterations:

1) Select a bootstrap set from the given training sample.
2) Grow/increase the un-pruned tree by using this bootstrap sample.
3) At every internal node, select m variables/predictors randomly and analyse best split using these predictors.
4) Do not perform pruning. Consider the tree as it has been build so far.

Estimate the result/output as overall prediction by taking the majority (classification) or average response (regression) from all the trees which are trained individually.

Advantages of Random Forest

- Random forest is simple classifier.
- It can be simply parallelized.
- It is resilient to noise and also for outliers.
- They provide evaluation of strength, error, variable importance and correlation.
- In random forest learning is very fast than other decision tree classification algorithm.
- They develop a classifier with very high accuracy for numerous datasets.
- It requires very few initial/pre-processing data.
- There is no need to select any variable before building of model starts.
- Random forest can be used to generate cluster identification using training sample proximity/relationship.

4.1.6 PART ALGORITHM

PART is based on divide and conquers strategy. PART algorithm produces decision lists. Decision lists are basically set of rules. New data is compared with each rule in the list and is assigns category according to first matching rule. A default category is assigned incase no rules in the list matches.

PART algorithm is a partial decision tree algorithm which is developed version of C4.5 and RIPPLE algorithm. The best thing in part algorithm is, there is no need to perform global optimization like j48 algorithm to generate the rules.

We can represent PART as a neural network which is known as PART neural network. Architecture consists of two layers:

1. Input layer (comparison layer) also called processing layer
2. Output layer (clustering, recognition layer)

The structure of this network is shown below in fig.4.1.6

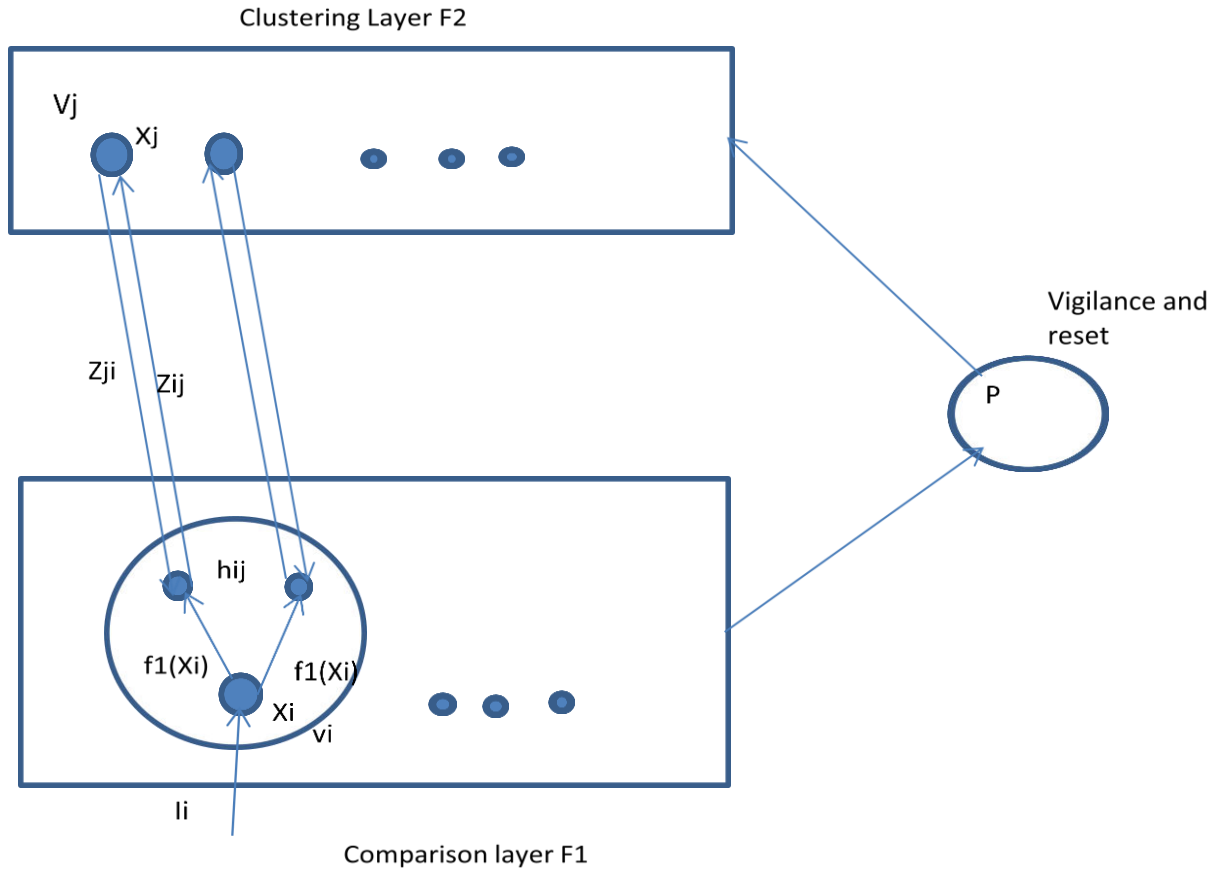


Fig 4.1.6 Architecture of PART Neural Network

From the figure, it can be seen that nodes in input layer (F_1) are represented by v_i, where i = 1, 2, 3,.....,m and nodes of clustering layer (F_2) are represented by v_j, where j = m+1, m+2,....,m+n. In this architecture two connections are available having their own weight. The first connection connects the neurons in F_1 layer to neurons in F_2 layer by weight w_{ij}, weight is known as bottom-up weight. Second connection is used to connect the neurons in F_2 layer to F_1 layer with weight $w_{ji,}$ weight is known as top-down weight. These connections may be changed according to the learning rule. h_{ij} which is the output signal sends signal only from node v_i in F_1 layer to node v_j in F_2 layer. Similarity check can be evaluated by using the function $f_1(x_i)$ produced in v_i and top-down weight (z_{ji}). Hence we can say that main focus of PART is only on those dimensions where information is available.

PART controls the degree of similarity by using two parameters which are independent of each other, distance σ and vigilance p. These parameters are used to control the degree of similarity and dimension's size. If PART has input with suitable range then it can helps us to find the number of clusters and the centre of each clusters.

4.1.7 LOGISTIC REGRESSION (LR)

Logistic regression is used to form prediction model and its quite similar to ordinary Least Square Regression. Whenever, we wish to use logistic regression an algebraic conversion is required to form usual linear regression equation. In case of logistic regression we do not get standard solutions and such solutions are not easily interpretable.

Logistic regression (LR) is a method which is used for prediction of change proneness (dependent variable) with the help of independent variables (object oriented metrics). These are used to determine how much variance present in the dependent variable that is expressed by the independent variables. The probability of an event to occur is predicted by placing the data in logistic regression curve. Logistic regression comes under statistical model which is used to deal with evaluation of mathematical models and showing the relationship between variables. Logistic regression is a standard technique which is based upon maximum likelihood evaluation for regression analysis. This model can be used to find the values of independent variables. To analyse the change proneness of class we need a prediction model and this model can be constructed by multivariate linear regression. Multivariate LR uses a group of metrics. Multivariate formula can be defined by following equation (univariate analysis is special case of this multivariate)

$$Prob\ (X_1, X_2, \ldots. X_n) = e^{(A_0+A_1X_1+\ldots..+A_nX_n)} / (1+e^{(A_0+A_1X_1+\ldots..+A_nX_n)})$$

where,

X_i, i=1,2,3,.....n are known as independent variables also called covariate.

Prob is the probability for finding a class has been changed or not. Prob is a dependent variable which is a conditional probability. Conditional probability is defined as the probability of failure/ fault found in a particular class.

The most general form of multivariate linear regression is represented as:

$$\hat{y_i} = a_0+a_1x_{i1}+a_2x_{i2}+a_3x_{i3}+\ldots\ldots\ldots+a_mx_{im}$$

$y_i = a_0+a_1x_{i1}+a_2x_{i2}+.....+a_mx_{im}+e_i$

where,

$x_{i1},x_{i2},.....x_{im}$ – independent variables

$a_0,a_1,a_2,.....a_m$- parameters which we have to estimate

$\hat{y_i}$ - dependent variable which we have to predict

y_i – dependent variable with actual value

e_i - error in prediction of i^{th} case

Logistic regression provides two stepwise selection methods:

1) Forward stepwise selection method
2) Backward stepwise elimination method

In forward stepwise selection method, we select a set of independent variables which is optimal. In every step of this model, we either add independent variables or delete it from regression model.

While in backward stepwise elimination method, it uses a model that adds all the independent variable. After adding, these are deleted one at a time till the converging/ stopping criteria is found.

Both of these methods do not guarantee that it will find the estimation criterion. Once we determine the independent variables that mean we have selected the model. And once the model is selected, the parameters a_0, a_1,.....,a_m can be estimated with the help of least square method. We use these estimated values of parameters to minimize error $\Sigma_{i=1}^{n} e_i^2$, where n is number of consideration/observations in the dataset.

4.2 PERFORMANCE EVALUATION MEASURES

4.2.1 Sensitivity: The total percentage of change prone classes that are correctly classified is known as sensitivity.

4.2.2 Specificity: The total percentage of change prone classes that are not correctly classified i.e. percentage of non-occurrences of correctly predicted classes.

4.2.3 Receiver Operating Characteristics (ROC): We evaluate the performance and accuracy of output of predicted model using area under curve (AUC). ROC curve (AUC) is defined as a plot of sensitivity on y-axis and 1- specificity on x-axis. The ROC curve, which is defined as a plot of sensitivity on the y-coordinate versus its 1-specificity on the x coordinate, is an effective method of evaluating the quality or performance of predicted models [19, 20]. While constructing ROC curves, we selected many cut-off points between 0 and 1, and calculated sensitivity and specificity at each cut off point. The optimal choice of the cut-off point (that maximizes both sensitivity and specificity) can be selected from the ROC curve [19, 20]. Hence, by using the ROC curve one can easily determine optimal cut-off point for a predicted model [12]. ROC curve can be drawn using SPSS tool.

4.2.4 Validation Method: In order to predict the accuracy of the model it should be applied to different data sets. We therefore performed a ten-cross validation of the models. The ten-cross is performed using WEKA tool. In this tool, complete dataset is divided into ten subsets randomly and each time one of the ten subsets is treated as testing set and other nine subsets are treated as training sets. Hence we obtain change proneness for all ten subsets.

CHAPTER 5

RESULT ANALYSIS

5.1 Univariate LR Results: Univariate is done as one to one correspondence. It is performed to examine significant and insignificant metrics. It is done by using SPSS tool and performed between one independent variable and one dependent variable. The same process is repeated of all metrics used. The same is performed for all software and we get univariate result for all software used. The univariate results of software ABRA, ABBOT, APOLLO, AVISYNC, JMETER are shown below in Table 5.1.1, 5.1.2, 5.1.3, 5.1.4 and 5.1.5.

In this section, we analyze significant and insignificant metrics based on metric sig. value. If sig. value of any is less or equal to 0.01 in three or more dataset then that metric is significant otherwise metric is insignificant. So from tables 5.1.1, 5.1.2, 5.1.3, 5.1.4 and 5.1.5, we can see that CBO metric is significant because its sig. Value for ABRA dataset is 0.000, ABBOT- 0.001, APOLLO- 0.000 and JMETER- 0.000. Similarly NIM, RFC, LOC, WMC are also significant metrics because the sig. value of all these metrics is less than or equal to 0.01 and remaining metrics are insignificant.

Table 5.1.1 Univariate Result for ABRA Dataset

Metrics	B	S.E.	Sig.	Exp(B)
CBO	.215	.047	**.000**	1.240
NOC	.152	.070	.031	1.164
NOM	-.462	.221	.036	.630
NOA	.071	.063	.256	1.074
NIM	.035	.015	.018	1.035
NIV	.043	.032	.181	1.044
NLM	.033	.015	.031	1.033
RFC	.027	.007	**.000**	1.028
NPRM	.021	.078	.790	1.021
NPROM	.109	.036	**.003**	1.115
NPM	-.024	.039	.535	.976
LOC	.002	.001	.046	1.002
DIT	.395	.204	.053	1.484
LCOM	.013	.006	.029	1.013
WMC	.010	.005	.050	1.010

Table 5.1.2 Univariate Result for ABBOT Dataset

Metrics	B	S.E.	Sig.	Exp(B)
CBO	-.017	.005	**.001**	.983
NOC	.000	.044	.995	1.000
NOM	.118	.038	**.002**	1.125
NOA	-.029	.023	.205	.972
NIM	-.011	.004	**.005**	.989
NIV	-.035	.010	**.001**	.966
NLM	-.008	.004	.027	.992
RFC	.002	.002	.426	1.002
NPRM	-.023	.007	**.001**	.977
NPROM	.091	.036	.011	1.096
NPM	.021	.010	.030	1.021
LOC	-.001	.000	**.003**	.999
DIT	.039	.090	.667	1.039
LCOM	.004	.003	.239	1.004
WMC	-.003	.001	.023	.997

Table 5.1.3 Univariate Result for APOLLO Dataset

Metrics	B	S.E.	Sig.	Exp(B)
CBO	.117	.029	**.000**	1.124
NOC	.005	.038	.904	1.005
NOM	.054	.066	.415	1.055
NOA	.054	.047	.253	1.055
NIM	.027	.015	.079	1.028
NIV	.042	.032	.190	1.043
NLM	.028	.015	.061	1.028
RFC	.040	.013	**.001**	1.041
NPRM	.022	.137	.871	1.022
NPROM	.543	.150	**.000**	1.722
NPM	.021	.016	.195	1.021
LOC	.001	.001	.108	1.001
DIT	.015	.192	.938	1.015
LCOM	.010	.004	.023	1.010
WMC	.019	.007	**.005**	1.020

Table 5.1.4 Univariate Result for AVISYNC Dataset

Metrics	B	S.E.	Sig.	Exp(B)
CBO	.142	.056	.012	1.152
NOC	-.168	.202	.405	.845
NOM	.129	.209	.538	1.137
NOA	.071	.067	.294	1.073
NIM	.136	.042	**.001**	1.146
NIV	.453	.146	**.002**	1.573
NLM	.138	.042	**.001**	1.148
RFC	.034	.020	.093	1.034
NPRM	.186	.096	.052	1.205
NPROM	-4.138	8038.594	1.000	.016
NPM	.147	.054	**.007**	1.159
LOC	.011	.004	**.008**	1.011
DIT	-.589	.225	**.009**	.555
LCOM	.004	.007	.597	1.004
WMC	.093	.030	**.002**	1.098

Table 5.1.5 Univariate Result for JMETER Dataset

Metrics	B	S.E.	Sig.	Exp(B)
CBO	.179	.020	**.000**	1.196
NOC	.212	.071	**.003**	1.236
NOM	-.038	.018	.032	.963
NOA	.040	.013	**.002**	1.040
NIM	.061	.010	**.000**	1.063
NIV	.140	.022	**.000**	1.150
NLM	.042	.008	**.000**	1.043
RFC	.013	.002	**.000**	1.013
NPRM	.154	.030	**.000**	1.167
NPROM	.061	.041	.140	1.063
NPM	.039	.009	**.000**	1.040
LOC	.004	.001	**.000**	1.004
DIT	.215	.059	**.000**	1.240
LCOM	.018	.002	**.000**	1.019
WMC	.016	.003	**.000**	1.016

From Table [5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5] which are presented in chapter 5, we can conclude the table explained below which shows the relationship between object oriented metrics and the software we have taken or we can say that this table 5.1.6 is the significance table.

Table 5.1.6 Relationship between Object Oriented Metrics and Software

	ABRA	**ABBOT**	**APPOLO**	**AVISYNC**	**JMETER**
CBO	Y	Y	Y		Y
NOC					Y
NOM		Y			
NOA					Y
NIM		Y		Y	Y
NIV		Y		Y	Y
NLM				Y	Y
RFC	Y		Y		Y
NPRM		Y			Y
NPROM	Y		Y		
NPM				Y	Y
LOC		Y		Y	Y
DIT				Y	Y
LCOM					Y
WMC			Y	Y	Y

From table 5.1.6, we can see that CBO, NIM, NIV, RFC, LOC, WMC metrics are the best significance metrics because these five metrics are significant in three or more than three software. The metrics NLM, NPRM, NPROM, NPM and DIT are the average significant metrics because they are significant in two software. And the metrics NOC, NOM, NOA, LCOM are the least significant metrics because these metrics are significant either only in one software or none.

5.2 MODEL EVALUATION USING ROC CURVE

Here we evaluate best model using ROC curve. The model having largest area under curve (AUC) will be the best model for a dataset. The tables below show the validation results on various datasets. Table 5.2.1 shows the validation result on ABRA dataset. Similarly Table 5.2.2, 5.2.3, 5.2.4 and 5.2.5 shows the validation result on ABBOT, APOLLO, AVISYNS and JMETER respectively. In table 5.2.1, NAIVE gave best result among all models having AUC 0.774, sensitivity and specificity are 0.727 and 0.728 respectively and its cut-off point is 0.054. In Table [5.2.2, 5.2.3], KSTAR shows the best result for both ABBOT and APOLLO software among all models having AUC 0.758, 0.779 respectively sensitivity and specificity are 0.709, 0.725 and 0.705, 0.723 respectively and cut-off point is 0.256, 0.112. Table 5.2.4 has MLP model as the best model which outperformed other method has AUC 0.783, sensitivity and specificity are 0.667 and 0.605 respectively and its cut-off point is 0.378. Table 5.2.5 shows that RANDOM FOREST (RF) is best model because it has AUC 0.831, sensitivity and specificity are 0.795 and 0.741 respectively and cut-off point is 0.568.

These best models of each dataset can be used to predict change prone classes. The advance knowledge of change prone classes would help us to plan the test resources for the classes in phase of software development process. These change prone classes needs to be allocated more resources than other non change prone classes because these classes needs to be tested many times. If a class is more change prone it means it needs greater effort in maintenance phase of software development. Thus if we will predict the change prone classes in beginning phases it will reduce the maintenance and testing efforts. Figure 5.2.1, 5.2.2, 5.2.3, 5.2.4 and 5.2.5 shows the ROC curve of best models of all five software.

Table 5.2.1 Model Evaluation Result of ABRA Dataset

Machine Learning Methods	**AUC**	**CUTOFF POINT**	**SENSITIVITY**	**SPECIFICITY**
Naïve Bayes	0.774	0.054	0.727	0.728
MLP	0.766	0.092	0.697	0.707
K-Star	0.611	0.065	0.545	0.565
Bagging	0.742	0.102	0.727	0.721
Random Forest	0.642	0.104	0.576	0.592
PART	0.726	0.075	0.697	0.667
LR	0.756	0.139	0.727	0.721

Table 5.2.2 Model Evaluation Result of ABBOT Dataset

Machine Learning Methods	AUC	CUTOFF POINT	SENSITIVITY	SPECIFICITY
Naive Bayes	0.608	0.426	0.581	0.577
MLP	0.656	0.270	0.593	0.593
K-Star	0.758	0.256	0.709	0.705
Bagging	0.755	0.245	0.674	0.676
Random Forest	0.707	0.162	0.686	0.647
PART	0.655	0.326	0.581	0.581
LR	0.675	0.266	0.593	0.589

Table 5.2.3 Model Evaluation Result of APOLLO Dataset

Machine Learning Methods	AUC	CUTOFF POINT	SENSITIVITY	SPECIFICITY
Naive Bayes	0.672	0.118	0.638	0.628
MLP	0.660	0.215	0.638	0.639
K-Star	0.779	0.112	0.725	0.723
Bagging	0.769	0.253	0.696	0.694
Random Forest	0.764	0.209	0.725	0.727
PART	0.721	0.299	0.638	0.656
LR	0.694	0.245	0.623	0.623

Table 5.2.4 Model Evaluation Result Of AVISYNC Dataset

Machine Learning Methods	AUC	CUTOFF POINT	SENSITIVITY	SPECIFICITY
Naive bayes	0.766	0.111	0.704	0.796
MLP	0.783	0.378	0.667	0.609
K-Star	0.771	0.414	0.704	0.713
Bagging	0.760	0.689	0.630	0.630
Random Forest	0.742	0.281	0.778	0.609
PART	0.721	0.375	0.630	0.630
LR	0.717	0.326	0.741	0.609

Table 5.2.5 Model Evaluation Result of JMETER Dataset

Machine Learning Methods	AUC	CUTOFF POINT	SENSITIVITY	SPECIFICITY
Naive Bayes	0.722	0.108	0.672	0.672
MLP	0.762	0.610	0.732	0.722
K-Star	0.797	0.629	0.758	0.755
Bagging	0.827	0.632	0.762	0.758
Random Forest	0.831	0.568	0.795	0.741
PART	0.790	0.654	0.708	0.736
LR	0.771	0.567	0.719	0.713

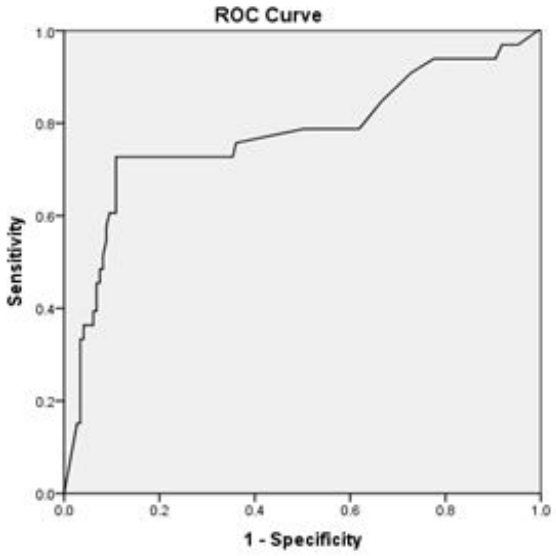


Fig 5.2.1 ABRA (Naive Bayes)

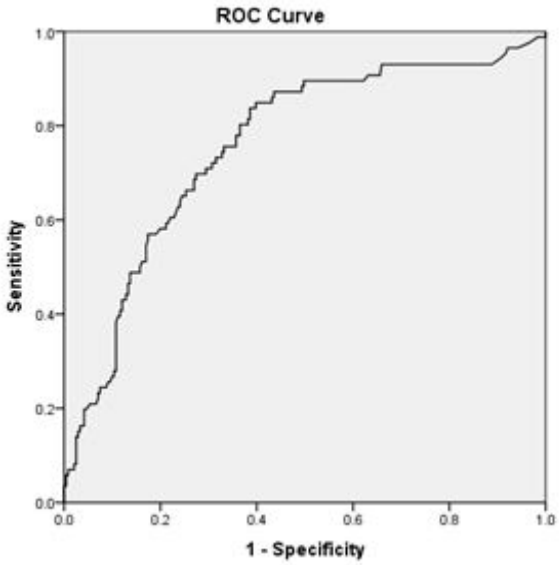


Fig 5.2.2 ABBOT (K-Star)

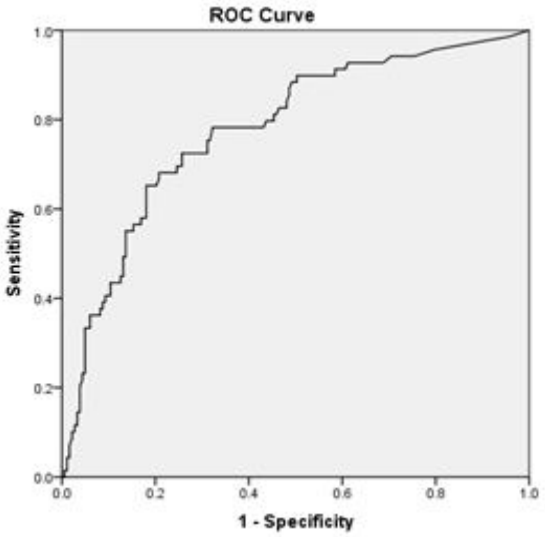


Fig 5.2.3 APOLLO (K-Star)

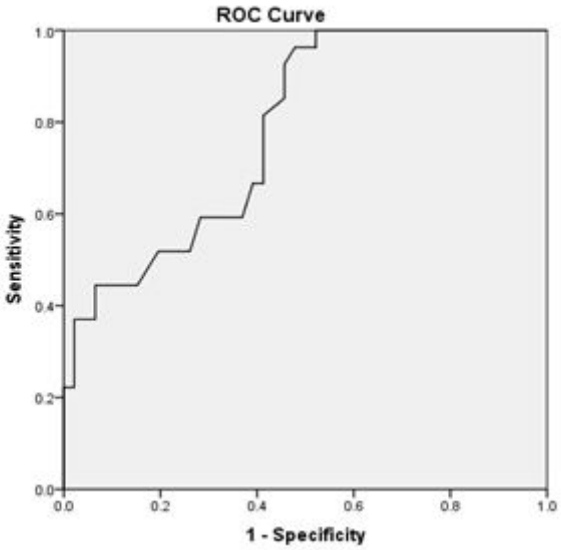


Fig 5.2.4 AVISYNC (MLP)

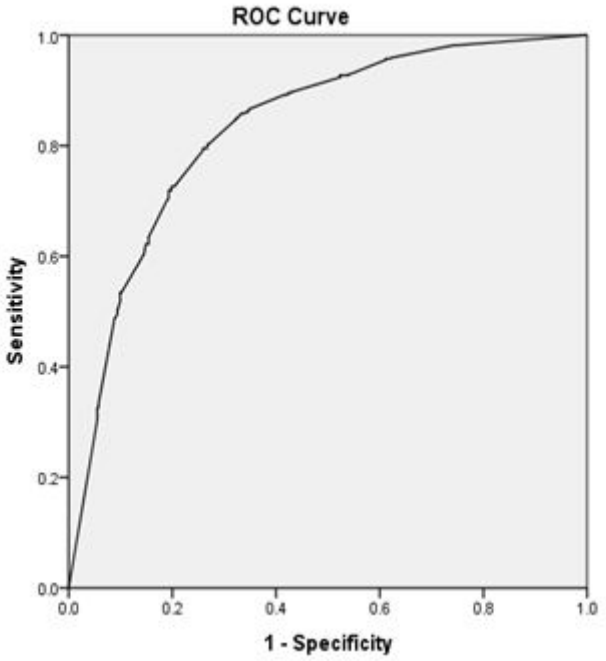


Fig 5.2.5 JMETER (Ramdom Forest)

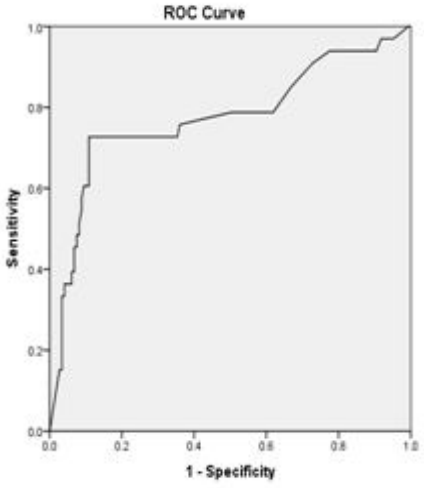
Table 5.2.6 Relationship between Software and Machine Learning Methods

	NAIVE BAYES	MLP	K-STAR	BAGGING	RANDOM FOREST	PART
ABRA	0.77					
ABBOT			0.758			
APPOLO			0.779			
AVISYNC		0.783				
JMETER					0.831	

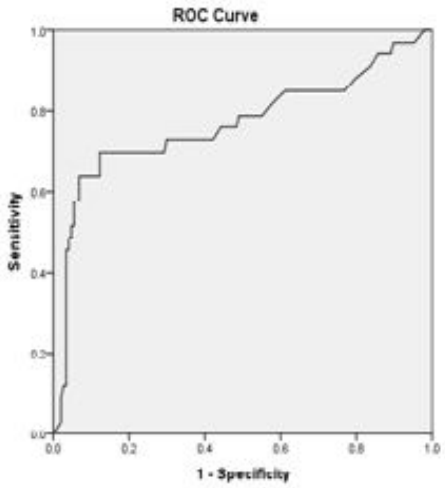
Table 5.2.6 shows the relationship between machine learning methods and software we have taken. We can analyze best model for each software from this table. So from the above table we can conclude that naive bayes method is the best model for ABRA software, K-STAR method is the best change proneness prediction model for two software namely ABBOT and APPOLO. Multilayer perceptron method is the best prediction model for AVISYNC software and RANDOM FOREST is the best model for JMETER software.

ROC Curves of Every Methodology of All the Software:

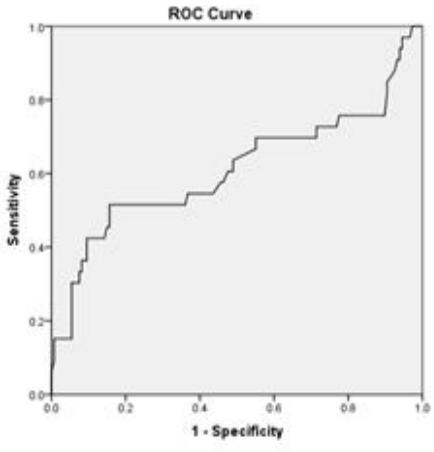
In this section we present receiver operating curve of every method like naive bayes, multilayer perceptron, k-star, bagging, random forest, PART machine learning methods for every software one by one. These are shown in figures presented below:



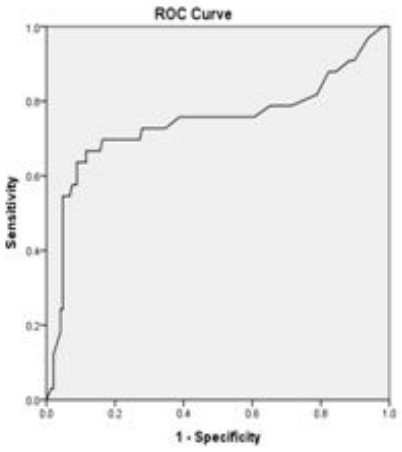
Naive Bayes



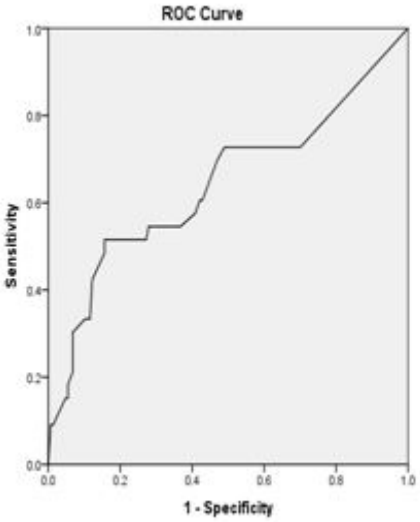
Multilayer Perceptron



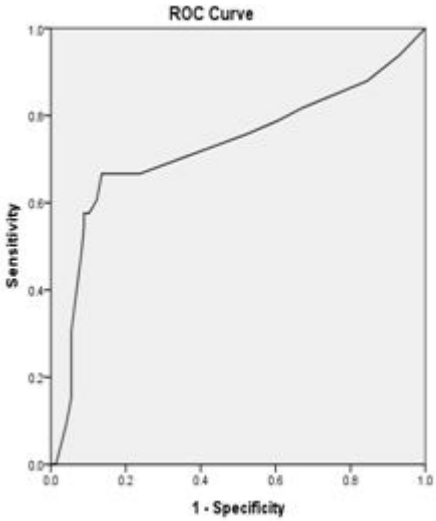
K-Star



Bagging

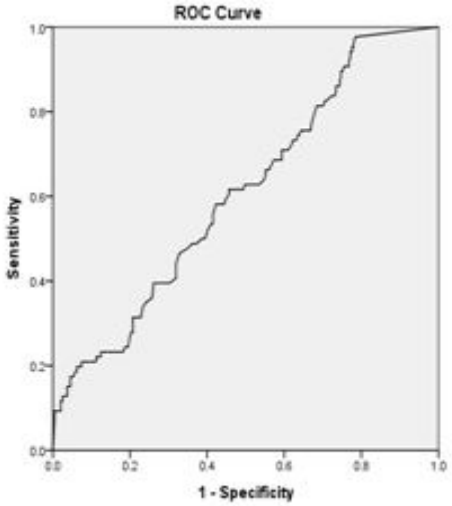


Random Forest

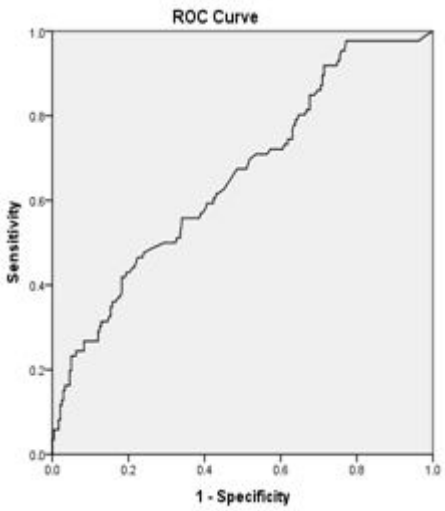


PART

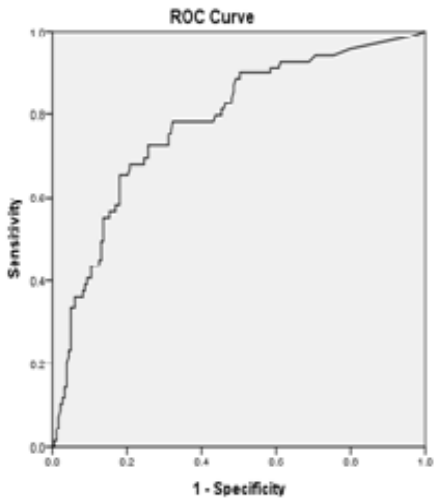
Fig 5.2.6 ROC Curve for ABRA Software



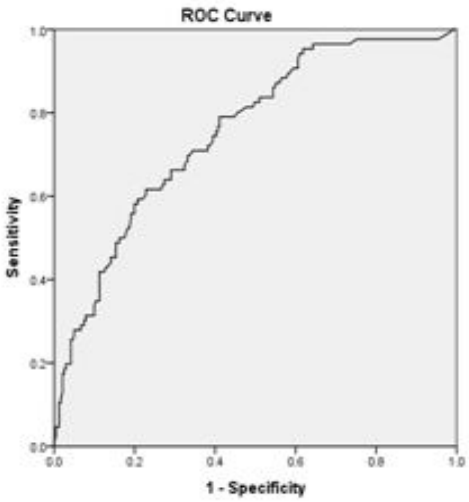
Naive bayes



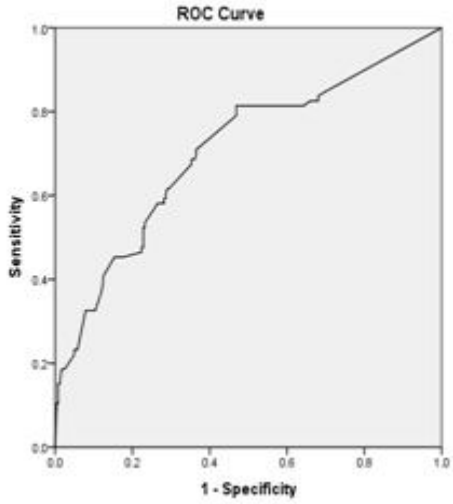
Multilayer perceptron



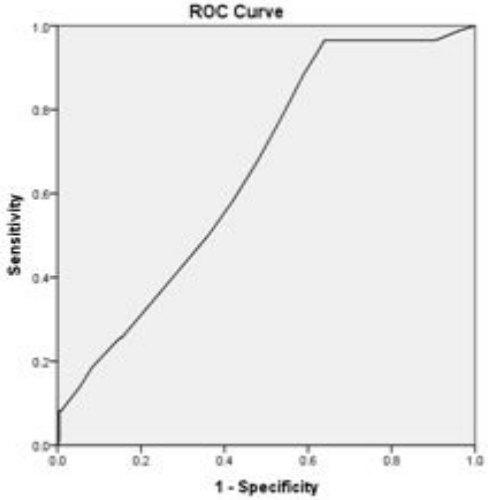
K-Star



Bagging

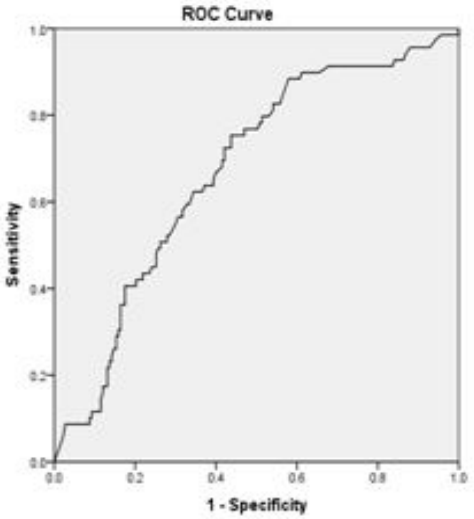


Random forest

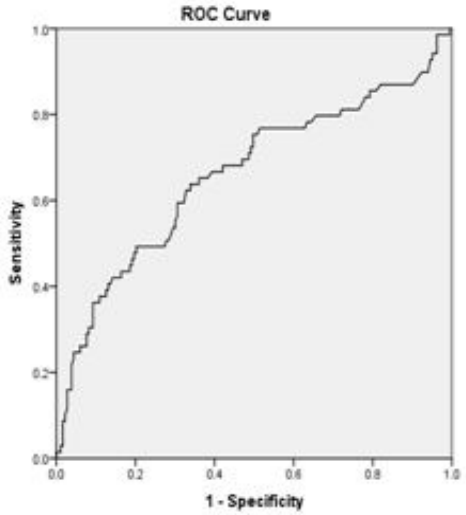


PART

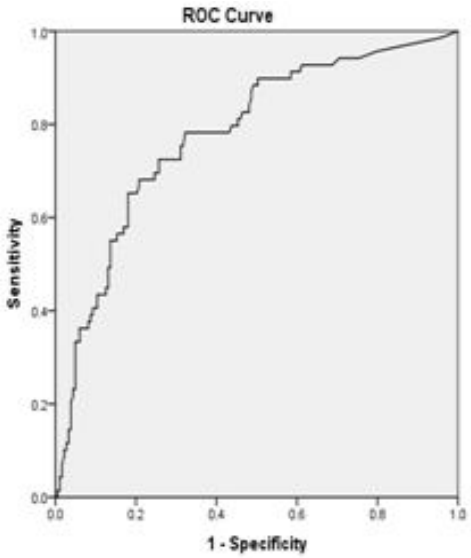
Fig 5.2.7 ROC Curve for ABBOT Software



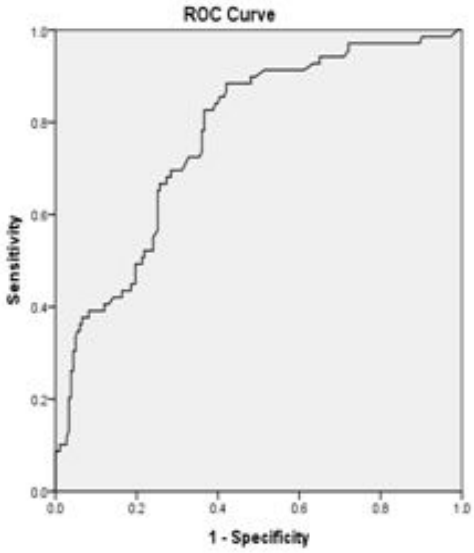
Naive bayes



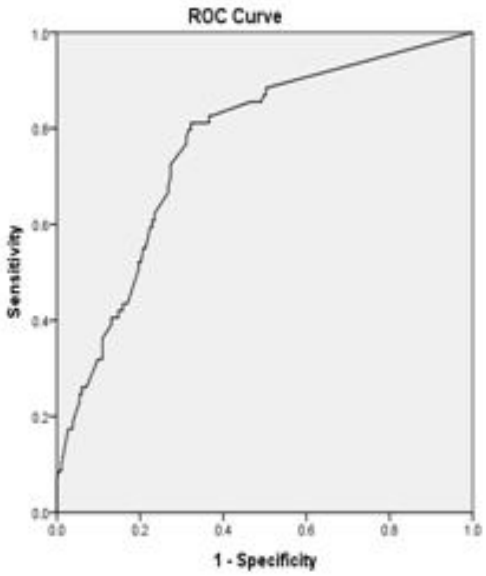
Multilayer perceptron



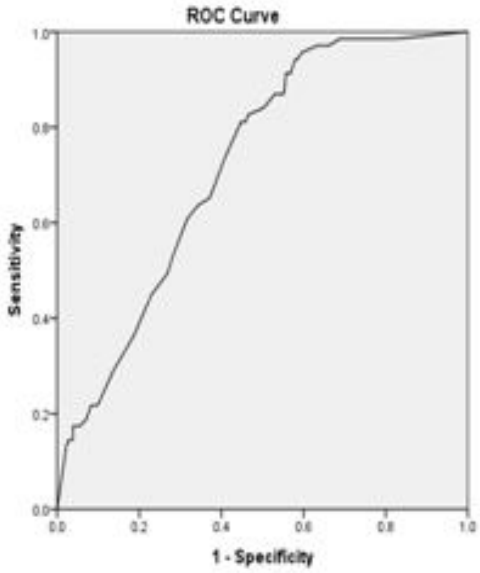
K-Star



Bagging

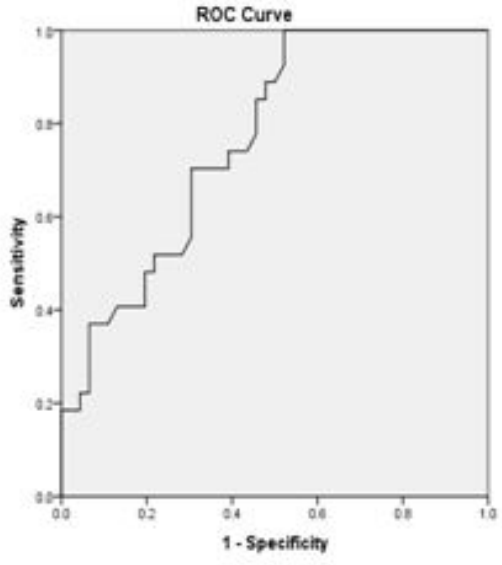


Random Forest

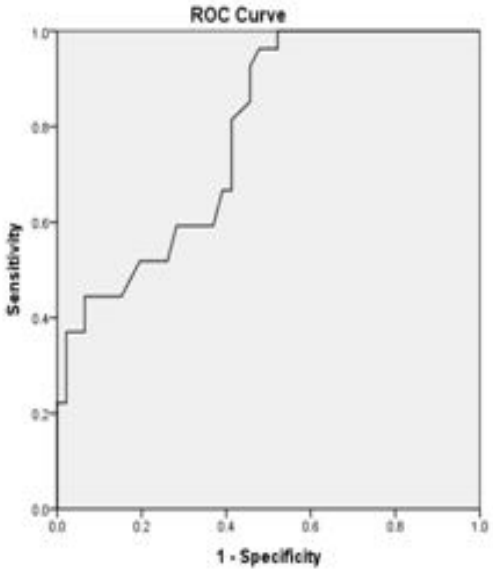


PART

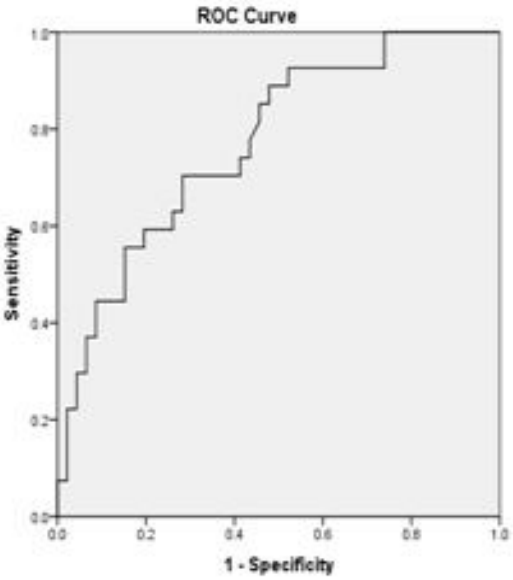
Fig 5.2.8 ROC Curve for APPOLO Software



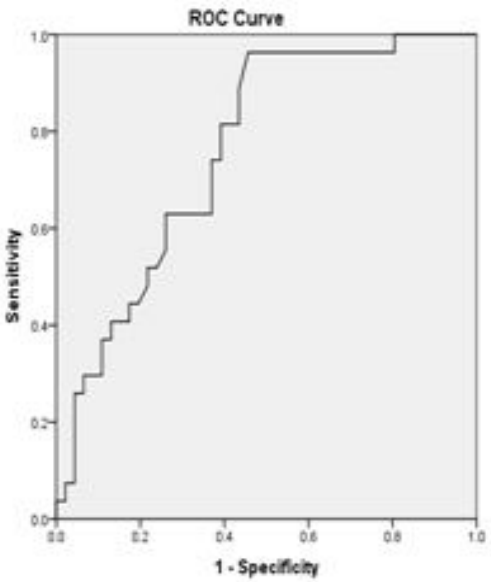
Naive bayes



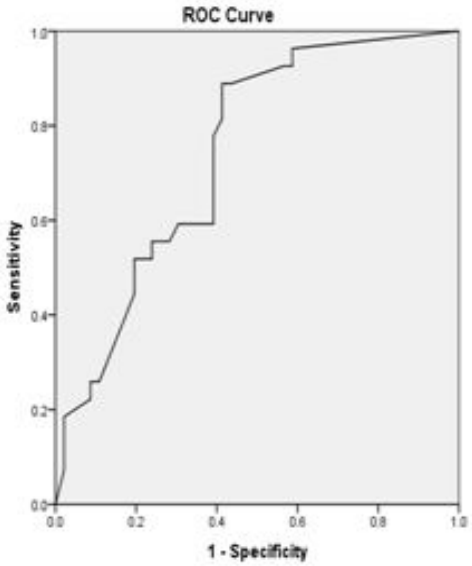
multilayer perveptron



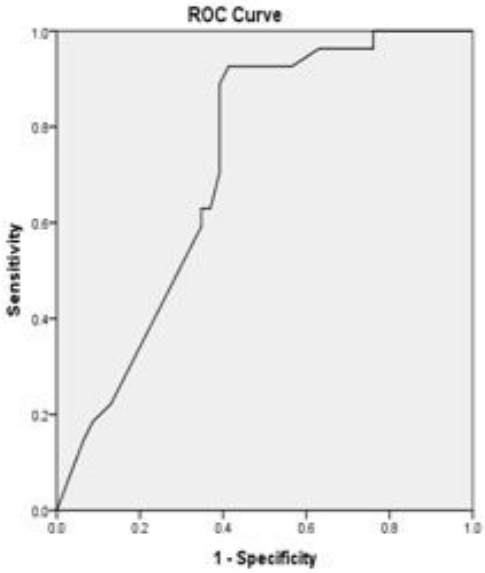
K-Star



Bagging

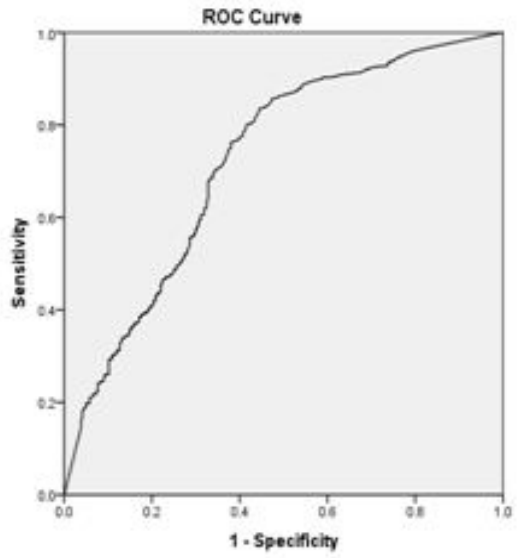


Random forest

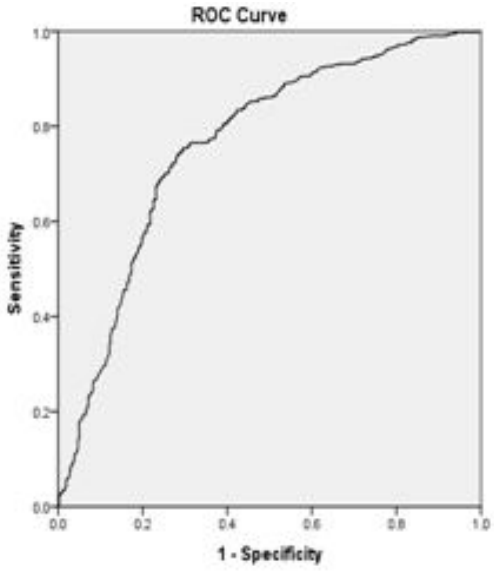


PART

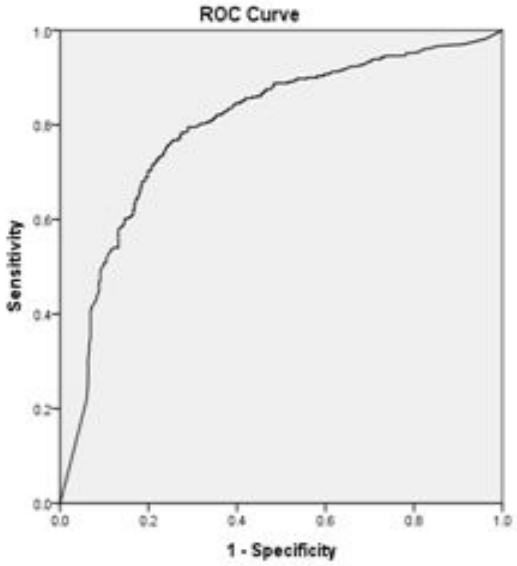
Fig 5.2.9 ROC Curve for AVISYNC Software



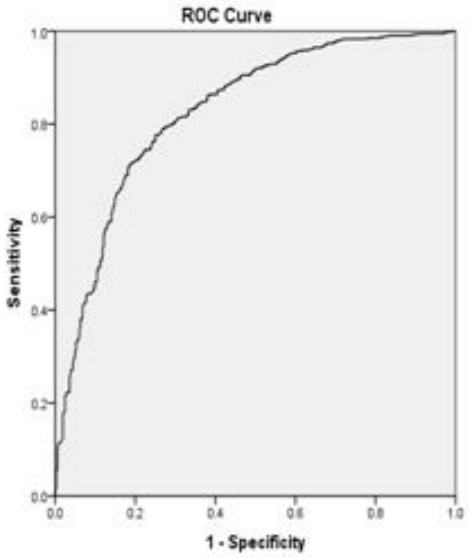
Naive bayes



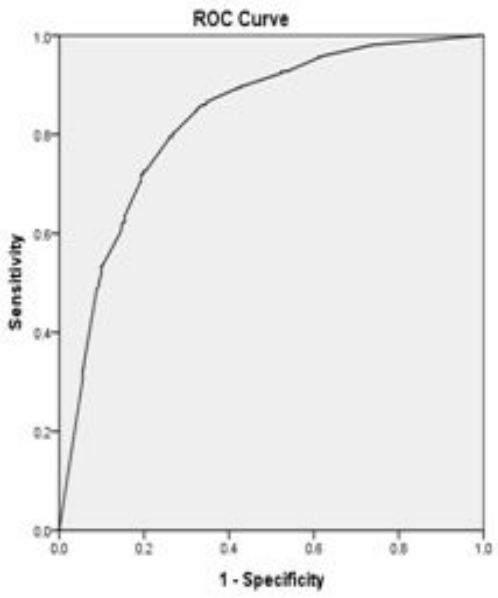
Multilayer perceptron



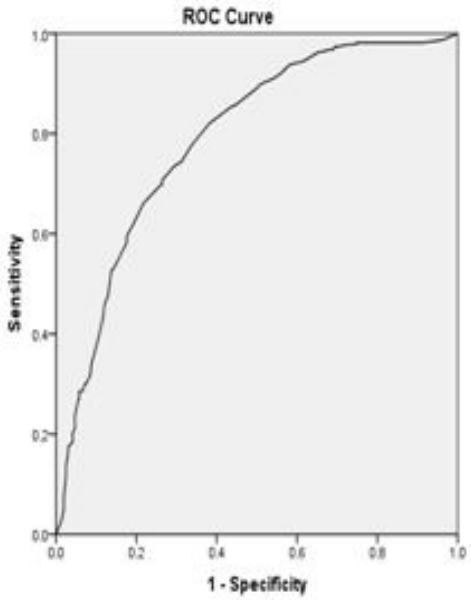
K-Star



Bagging



Random forest



PART

Fig 5.2.10 ROC Curve for JMETER Software

CHAPTER 6

CONCLUSION AND FUTURE WORK

Our main objective of this study is to evaluate the relationship between change proneness and object oriented metrics for a class. We analyze and compare machine learning methods, logistic regression method for performance evaluation which will help us in prediction of change proneness of classes. As we have shown that in our study we have used five software namely ABRA, ABBOT, APPOLO, AVISYNC, JMETER and also obtained dataset for these software and evaluated performance of these software on the basis of receiver operating curve analysis.

The main results are shown below:

1. CBO, NIM, NIV, RFC, LOC, WMC metrics are the best significance metrics because these five metrics are significant in three or more than three software. The metrics such as NLM, NPRM, NPROM, NPM and DIT are the average significant metrics because they are significant in two software. And the metrics NOC, NOM, NOA, LCOM are the least significant metrics because these metrics are significant either only in one software or none.

2. From our study, we can conclude that Naive Bayes method is the best method for ABRA software, K-STAR method is the best method for change proneness prediction model corresponding to two software namely ABBOT and APPOLO. Multilayer perceptron method is the best method for AVISYNC software and RANDOM FOREST is the best method for JMETER software.

3. The above results can be used in initial phases of software development to reduce the cost and maintenance effort once the software has been delivered to the customer. Also the best models predicted from our study can help us to plan test resources efficiently and effectively.

In future, we intend to replicate our study on similar datasets in order to get more generalized results. In this study we have used WEKA and SPSS tool but in future we will try to apply one more tool named KEEL. KEEL is a tool that includes all the evolutionary methods such as Genetic Algorithm, fuzzy logic, evolutionary neural network etc. evolutionary methods are

the best methods and have many advantages such as: 1. We can apply evolutionary algorithms to that problem where no good method is available, for example discontinuities problem. 2. Somewhere, we required multiple solutions so in that case also we can use evolutionary methods. 3. Parallel implementation of these algorithms are also easy.

ABBRAVIATIONS

Serial number	Keywords	Description
1	CBO	Coupling between object
2	NOC	Number of children
3	NOM	Number of class methods
4	NOA	Number of class variables
5	NIM	Number of instance methods
6	NIV	Number of instance variables
7	NLM	Number of local methods
8	RFC	Response for a class
9	NPRM	Number of local private methods
10	NPROM	Number of local protected methods
11	NPM	Number of local public methods
12	LOC	Lines of code
13	DIT	Depth of inheritance tree
14	LCOM	Lack of cohesion in methods
15	WMC	Weighted method per class
16	LR	Logistic regression
17	MLP	Multilayer perceptron
18	RF	Random forest
19	NB	Naive bayes
20	RFC	Receiver operating curve
21	AUC	Area under curve
22	WEKA	Waikato Environment for knowledge analysis
23	SPSS	Statistical package for the social science
24	OO	Object oriented

REFERENCES

[1] AR Han, S Jeon, D Bae, J Hong (2008) "Behavioral Dependency Measurement For Change Proneness Prediction In UML 2.0 Design Models", in computer software and applications 32nd annual IEEE international.

[2] M D'Ambros, M Lanza, R Robbes (2009) "On the Relationship between Change Coupling and Software Defects" in 16th working conference on reverse engineering, pp 135–144.

[3] AR Sharafat, L Tavildari (2007) "Change Prediction in Object Oriented Software Systems: A Probabilistic Approach" in 11th European conference on software maintenance and reengineering.

[4] AM Chaumum, H Kabaili, RK Keller, F Lustman (1999) "A Change Impact Model For Changeability Assessment in Object Oriented Software Systems" in third european conference on software maintenance and reengineering, pp 130.

[5] Y Zhou, H Leung, B Xu (2009) "Examining the Potentially Confounding Effect of Class Size on the Associations between Object Oriented Metrics and Change Proneness" IEEE Trans Software Eng 35(5):607–623.

[6] J Bieman, G Straw, H Wang, PW Munger, RT Alexander (2003) "Design Patterns and Change Proneness: An Examination of Five Evolving Systems" In: The proceeding of 9th international software metrics symposium.

[7] N Tsantalis, A Chatzigeorgiou, G Stephanides (2005) "Predicting the Probability of Change in Object Oriented Systems" IEEE Trans Softw Eng 31(7):601–614.

[8] Mikael Lindvall (1999) "Monitoring and Measuring the Change- Prediction Process at Different Granularity Levels" Software Process Improvement and Practice, vol.4, no. 1, pp.3-10.

[9] Ruchika Malhotra, Megha Khanna (2012) "Investigation of Relationship between Object-Oriented Metrics and Change Proneness" Springer-Verlag.

[10] Claire Ingram, Steve Riddle (2012) "Using Early Stage Project Data to Predict Change-Proneness" 978-1-4673-1762-7/12c IEEE.

[11] Daniele Romano, Paulius Raila, Martin Pinzger, Foutse Khomh "Analyzing the Impact of Antipatterns on Change-Proneness Using Fine-Grained Source Code Changes" 1095-1350/91.

[12] Ruchika Malhotra, Akshit Peer (2013) "Application of Adaptive Neuro-Fuzzy Inference System for Predicting Software Change Proneness" 978-1-4673-6217-7/13 IEEE.

[13] Ruchika Malhotra, Ankita Jain (2012) "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality" Journal of Information Processing Systems, Vol.8, No.2.

[15] Ruchika Malhotra, Megha Khanna (2013) "Inter Project Validation for Change Proneness Prediction using Object Oriented Metrics" An International Journal (SEIJ), Vol. 3, No. 1.

[16] Ruchika Malhotra, Anushree Agrawal "CMS Tool: Calculating Defect and Change Data from Software Project Repositories" DOI: 10.1145/2557833.2557849.

[17] L Briand, J Wust, J Daly, DV Porter (2000) "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems" J Syst Softw 51(3):245–273.

[18] SR Chidamber, DP Darcy, CF Kemerer (1998) "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis" IEEE Trans Software Eng 24(8):629–639.

[19] M Cartwright, M Shepperd (2000) "An Empirical Investigation of an Object-Oriented Software System" IEEE Trans Softw Eng 26(8):786–796.

[20]. L Breiman (1996) "Bagging Predictors" Mach Learn 24:123–140.

[21] S Bengio (2006) “Statistical Machine Learning from Data Ensembles” IDIAP Research Institute, Martigny, January 27.

[22] Ruchika Malhotra, Anikta Jain (2011) “Software Effort Prediction using Statistical and Machine Learning Method,” International Journal of Advanced Computer Science and Applications , Vol.2, No.1.

[23] Tom M. Mitchell (2010) “Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression”,

[24] R. Krakovsky, I. Mokris (2012) “Clustering of Text Documents by Projective Dimension of Subspaces using PART Neural Network” 978-1-4673-1014, IEEE.

[25] Cuiping Leng, Shuangcheng Wang, Hui Wang (2009) “Learning Naive Bayes Classifiers with Incomplete Data” 978-0-7695-3816-7/09, IEEE, DOI 10.1109/AICI.2009.402.