ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

A MODEL DEVELOPMENT FOR SOFTWARE DEFECT PREDICTION: USING FEATURE REDUCTION METHOD

ABAS AMAN

February, 2018

Adama, Ethiopia

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

A MODEL DEVELOPMENT FOR SOFTWARE DEFECT PREDICTION: USING FEATURE REDUCTION METHOD

BY

ABAS AMAN

The Thesis Submitted to School of Graduate Studies of Adama Science and Technology University for Partial Fulfillment of the Requirements for the Degree of Master of Science in Information System

February, 2018

Adama, Ethiopia.

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DEPARTMENT OF COMPUTING

A MODEL DEVELOPMENT FOR SOFTWARE DEFECT PREDICTION: USING FEATURE REDUCTION METHOD

BY

ABAS AMAN

SIGNATURE PAGE (Approval sheet)

Name of Student	**Signature**	**Date**
Shimelis Assefa, Ph.D.	Shimelis Assefa	December 21, 2017
Advisor	**Signature**	**Date**
Head of the Department	**Signature**	**Date**
School Dean	**Signature**	**Date**
SGS Dean	**Signature**	**Date**

DECLARATION

This is my original work and has not been submitted for a degree in any university.

Name: Abas Aman

Signature_____Date _____

Acknowledgement

First of all “Alhamdulillah”. Thanks to my Almighty Allah for the help to come up at this stage.

In this thesis, I had received valuable assistance from various professionals, my friends and my family. If this is the case, I have to express my heart-felt gratitude to those people who deserve my acknowledgment.

The second and top thanks would goes to my advisor, Dr.Shimiles Assefa at University of Dendever for his guidance, valuable advice; critical comments and constructive suggestions that are the major elements for the completion of this thesis.

My special thanks to Dr. Mesfin Abebe, the coordinator of PG study in the department of computing for his basic and concrete suggestion and comments on machine learning area from scratch for my thesis.

I am also indebted to give other special thanks to Ermias Berhanu (MSc) Lecturer at Wolkite University. He has helped me as co-advisor from the starting up to the end, especially how NASA raw data can change from different format to the machine learning file format. In addition to this, I never pass without express my deepest thanks to Mohammed Kemal (Head of the Department), Dr.Nirmila, and Musa Dima (PhD followers in Turkish, at Trapzon University) those who initiated me in this area at the proposal level.

I would like to thank the NASA MDP Organization for making their defect data sets publicly available and WEKA team for freely available open source for so many researchers like me.

My heartfelt also goes to my friends, Hassan Hussen, Sultan Ermias, Mohammed Tukie, Ali Jabo (my best friend) and Mohammedamin Jemal MSc candidate at Oromia State University, my elder brother Abdulhakim Aman who lives in Doha, Amu Tarik, Dr.Ahmed Kelil and all my family’s in general for their supportive in all things.

At the last but not the least, I NEVER FORGET to acknowledge my LOVELY wife Ifnan Mohammed for her support and encouragement for the accomplishment of this paper. I just want to say “I LOVE YOU MORE I CAN SAY”.

Table of Contents

Lists of Tables

Lists of figures

Lists of Equations

ACRONYMS

AI	**Artificial Intelligence**
ARFF	**Attribute Relation File Format**
DVD	**Digital Versatile Disk**
CK	**Chidambaram and Kemmerer**
GUI	**Graphic User Interface**
IEEE	**Institute of Electrical and Electronics Engineers**
IT	**Information Technology**
LOC	**Line of Lode**
MDP	**Modular toolkit for Data Processing**
ML	**Machine Learning**
NASA	**National Aeronautics and Space**
NIST	**National Institute of Standards and Technology**
ROC	**Receiver Operating Characteristic**
SDLC	**Software Development Life Cycle**
SDP	**Software Defect Prediction**
SVM	**Support Vector Machine**
WEKA	**Waikato Environment for Knowledge Analysis**

ABSTRACT

Software is a significant element in many of the devices and systems that pervade our today's societies. Many software companies are building different size of software systems for various purposes to deliver the software within short time and better quality to their customers. Due to the increasing size of software data and the constraints under which the software is developed, it is too difficult to produce quality software in shorter time. Software defect is errors that are introduced by software developers (testers) and stakeholders through testing. Defect can be error, fault, bug or failure. Therefore, defect prediction before delivering the software product can contribute significantly to the success of a project in terms of; quality and cost. Software companies have various measurement methods to test their software quality.

The goal of this study is to apply different software data set before preprocessing and feature reduction in machine learning algorithms or methods for predicting the status of software defect from the NASA MDP data set. Especially, the researcher has gave emphasize on a model development for software defect prediction using feature reduction method plus prototype system for identification of faults.

There were some methodologies, including initial literature review up to evaluation of the performance and developing the proposed model. The researcher gave more attention for both before preprocessing and after feature reduction of the machine learning approach by using four data set,CM1,KC1,KC2,PC1. Machine learning algorithms such as NBTree, Logistic and NB are used for the experimentation through Weka machine learning tool to build the prediction model. The model accuracy is tested using the 10-fold cross validation mode and prest software tool has been used to show the extraction of software features.

The best performing methods is identified by comparing their accuracy, specificity and execution time of build the models. Based on performance evaluator the best algorithm was found to be the(NBT) classifier, in PC1 data set feature reduction the researcher was gets 93.5% accuracy from the experiment. As a result the researchers identified that the feature selection should be tested before different software data set testing or maintaining taken place by testers or developers.

Keywords:- Software Defect Prediction, Feature Selection, Software Metrics, Machine Learning, Model, Data set, Preprocessing, Performance, Accuracy.

CHAPTER ONE

INTRODUCTION

1.1. Background of the study

Software is a significant element in many devices and systems that pervade our today's societies. Software defines an essential component of surrounded applications that control striking applications such as airplanes, spaceships, and air traffic control systems, as well as everyday appliances such as watches, stoves, cars, DVD players, garage door openers, cell phones, remote controllers and the behavior of network routers, financial networks, and telephone switching networks, web, and other infrastructure of modern life. It means that, Software's are used almost everywhere and in every tread of our life.

As a result, In order to fulfill the necessity of software, within short time and better quality to their customers or end users, many software companies are building different huge systems rapidly for various purposes, alongside with these reasons, as our dependency on software is increasing; software quality is becoming gradually more and more important in present era. On the other hand, Software consequences such as, fault and failures may diminish the quality of software which leads to customer dissatisfaction. Software that does not work correctly can lead to many problems, including loss of money, time or business reputation and could even cause injury or death.

The National Institute of Standards and Technology (NIST) has estimated that software defects and problems annually cost 59.5billions dollars the U.S. economy and approximately about 36% of the IT budget is spend on bug fixing. (Geralds, 2002).In addition to this, the number of defect's problem rapidly increases in relation to closed problems at the service counter; and customers have to wait a long time for solutions to problems. So defects in software product and process are cause much loss of resources (time and money).

To provide or in order to serve the quality software for users, by any means software testing is must. Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is defect) and to evaluate the features of the software item under software testing (IEEE, 1990). Faults (defects) or fault-proneness of software modules are to be

predicted in the early stages of software life cycle, so that more testing efforts can be put on faulty modules (Gayathri M, A. Sudha, 2014).

In today's world the most active research areas in software engineering is Software defect prediction. Software Development Process (SDP) or Software life cycle is a human activity, so it is unmanageable to avoid the injection of defects but it is possible to produce the software with few defects (Naheed Azeem,Shazia Usman, 2011).Typically the quality of the software could be measured by the number of faults or bugs that are searched in software products. Software defect is errors that are introduced by software developers (testers) and stakeholders through testing. Defect can be error, fault, bug or failure. Various metrics in software like cyclomatic complexity, Lines of Code have been calculated and effectively used for predicting faults.

The use of software fault classification is thus, to prevent faults and find as many faults, as early as possible. To deliver defect free software it is imperative to predict and fix the defects as much as possible before the product is released to the customers in testing life cycle. But, the problem is here, as software data set increases the time taken, number of tester experts as well as cost for testing increases. Due to the increasing size of software data set and the constraints under which the software is developed, it is too difficult to produce quality software in shorter time. Therefore, defect prediction before delivering the software product can contribute significantly to the success of a project in terms of; quality and cost. To come up with this solution, applying machine learning is desirable by developing a model to make easy a huge software data for developers in order to test.

Machine Learning is one of the most vital and motivating area of research with the objective of finding meaningful information from huge data sets (D.Tomar and S.Agarwal., 2013)**.**There are two most common category of machine learning supervised and unsupervised. For this study the researcher concerned only the supervised one. In machine learning, feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Additionally, given a dataset, in feature selection can be generalized as the process of selecting a subset of features for use in further data analysis. This selected subset of features is expected to capture maximum information present in the dataset, i.e. the selected feature subset should contain the most prominent features for model construction. In many real life systems, feature selection is very important in identifying the behavior and performance of the system. Especially in biomedical applications, feature selection can play a pivotal role in identifying biomarkers.

According to (Wahono, 2015) research analysis of most relevant studies current software defect prediction researches concentrate on five topics and trends: estimation, association, classification, and clustering and dataset analysis as a core of machine learning. The total distribution of defect prediction methods are as follows: about 77.46% of the research studies are related to classification methods, 14.08% of the studies focused on estimation methods, 5.63% of the primary studies are datasets analysis topics , and 1.41% of the studies concerned on clustering and association methods. In addition, 64.79% of the research studies used public datasets and 35.21% of the research studies used private datasets.

In software defect prediction, various types of data from different repository are used in order to determine software defects labels. In this study, the researcher apply different machine learning techniques on several publicly available datasets of the National Aeronautics and Space Administration(NASA) software repository such as CM1, KC1, KC2 and PC1.The determination is to classify the software modules into either defect prone or not defect prone modules through three different machine learning techniques such as NBTree, Logistic and NB were used in this paper. In addition to this software metrics, prest tool has also used to show how to extract feature from software code.

Feature selection is the process of identifying and removing as much irrelevant and redundant information as possible, this reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. As the aim of the researcher is a model development for software defect prediction: using feature reduction method.

1.2. Software Miscarriage Histories

As today's software grows rapidly in size and density, software reviews and testing play a crucial role in the software development process, especially in capturing software defects. Unfortunately, software defects or software faults are very expensive activities in cost. (Jones,C.,& Bonsignour,O, 2012) had reported that the cost of finding and correcting defects is one of the most expensive software development activities. The cost of software defect increases over the software development stage. During the coding stage, capturing and correcting defects costs $977 per defect. The cost increases to $7,136 per defect in the software testing phase, then, in the maintenance phase, the cost to capture and remove increases to $14,102 (Boehm,B.,& Basili,V.R., 2001). In addition to this, the impact of Software defect cost huge money for companies that develop code. A well-

publicized study ("The Economic Impacts of Inadequate Infrastructure for Software Testing," in May 2002 ,by the National Institute of Standards and Technology (NIST), estimates that software defects (Shichaozhang,C.Z.aq, 2003) are cost the US economy $60 billion annually. The study also claims that $22 billion of yearly cost could be eliminated by improved testing procedures early in the software development process. The economic impacts of software defects have made software quality an important issue in both industry and academia. In addition to economical costs, so that one could argue software defects impose negative social effects on both software users and software developers (Lieberman, 1997). As far as the researcher reviewed many literatures, there is no any report similar to the above in Ethiopian context.

1.3. Motivation for the Study

Learning from past experience, it would be possible to predict defects in advance for new software products. Finding and fixing the defects after delivery usually consumes a large portion of the project budget. The aim of this research study is to explore the different issues and problems in the area of defect prediction as well as provide the solutions by feature selection for software defect to improve software product quality and performance of its defect prediction. This motivates the researcher to find a way to minimize the cost and time usage, then the solution come in concentration; is that developing a model for software defect prediction: using feature selection method ,after that ,use these model for any software development before testing and implementation.

1.4.Statement of the Problem

It is a common practice that any software and any service product before delivered to customer is measured or tested by security, reliability, error free, and fault tolerance level. Among this, Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item. Additionally, Software testing is an activity that should be done throughout the whole development process. The ability to measure software defect can be extremely important for minimizing cost and improving the overall effectiveness of the testing process. The major amounts of defects in a software system are found in a few of its components. (Sahana D. C., 2013) It is a fact that, a composite software application with many bugs would be perceived as a poor-quality product. Unfortunately, large software systems data set tend to contain a lot of defects (Zheng, 2010).Therefore, software quality

assurance is still a challenging problem for software development industries, specifically for the project managers and testing departments.

According to (Zheng, 2010) suggested numerous branches from errors and mistakes made by programmers, whereas a small number of defects are caused by compilers that produce incorrect code, in the design and coding process. These defects not only reduce the quality of software but also drive up the cost of testing (Whittaker,J, 2000).Definitely, Almost any software development organizations have expended a huge amount of money and effort on testing of their software products before releasing them to their customers, even including the known Microsoft. In order to well manage software defects, additional human resources need to be hired as software testers. In many companies, the tester to developer ratio is even 1:1 (Cem Kane,J. D.,Hendrickson, E.,& Smith-Brock,J, 2001). However, since complete testing is very expensive and infeasible given the finite budget and personnel resources, many final products still contain defects. It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that software is 100% tested. The current Defect prediction mechanism has no testing of the effective feature selection to predict software as defect and no defect. Researchers are not capable to identify a generalized subset of attributes selection which acts as a considerable factor for a module to be defect or non-defect. (N.Gayatri,S.Nickolas,A.V.Reddy, 2010) Data for defect prediction is available in large extent from the data sources.

One of the problems with huge software data sets is high dimensionality, which means numerous unwanted and irrelevant features are present which the full picture of inaccurate, unexpected and redundant outcome of the model.

The majority of real-world classification problems require supervised learning where the underlying class probabilities and class-conditional probabilities are unknown, and each instance is associated with a class label (M.Dash and H.Liu, 1997). In real-world situations, we often have little knowledge about relevant features. Therefore, to better represent the domain, many candidate features are introduced, resulting in the existence of irrelevant or redundant features to the target concept. A relevant feature is neither irrelevant nor redundant to the target concept; an irrelevant feature is not directly associate with the target concept but affect the learning process, and a redundant feature does not add anything new to the target concept (M.Dash and H.Liu, 1997).In many classification problems, it is difficult to learn good classifiers before removing these unwanted features due to the

huge size of the software data. Reducing the number of irrelevant or redundant features can drastically reduce the running time of the learning algorithms and yields a more general classifier. This helps in getting a better insight into the underlying concept of a real-world classification problem. Sometimes irrelevant features may lead to complexities in classification, so these irrelevant features or attributes must be eliminated as to get the best outcome performance classification.

Therefore, feature selection techniques have to be proposed in addition to developing a model for software defect prediction. The above mentioned problems are the most critical from the point of view of the researcher and that must be solved by this study.

The following listed are the research questions this study attempt to answer:-

Question 1: What is the impact of feature selection on software defect prediction performance?

Question 2: Which machine learning technique shows the top defect prediction performance by feature selection?

Question 3: What is the relation between software defective factors and feature selection in software defect prediction process?

1.5. The Objectives of the Study

1.5.1. General Objective

The major objective of this study is to design and develop a model for software defect prediction using feature reduction technique in software testing to improve the accuracy and performance of the prediction model.

1.5.2 Specific objective

The specific objective of this study includes the following:-

- Investigate the current software defect predictions and software metrics.
- Prepare data sets in the form of suitable format for machine learning algorithms.
- To evaluate the current state of defect prediction attributes selection and model.
- Apply selected machine learning techniques and analysis the experiment results.
- Develop software attribute reduction predictive model
- Develop a prototype for the model.

- Eventually put the conclusion and future work.

1.6. Scope and limitation of the Study

In this thesis the scope of the study has focused on software feature selection for defect prediction at code level to design and develop model for software defect prediction in software testing which classify defective instances with non-defective ones. The data sets used had taken from public NASA Promise Modular toolkit for Data Processing (MDP), such as, CM1, KC1, KC2 and PC1 only, An analysis together with explanation of the produced result and Measure the performance, Accuracy through precision and recall had analyzed here. The researcher only considered the software product metrics in feature selection in code. The limitations of this study are the measuring and evaluating prediction performance tool (the researcher are only using the performance measurement tool in WEKA software), and the researcher is not going to apply software metrics in each software, instead the researcher used the data sets that are already measured by software metrics tools and provided for research purpose by NASA to develop and test Performance of model.

1.7. Research Methodology

This study is experimental type; the datasets that the researcher has been analysis and understood by software product metrics and build a model with the selected machine learning techniques, by attributes selection, finally measure the performance and Accuracy of the best model. To come up with the determination the above objectives , the following methodologies are used accordingly.

1.7.1. Literature Review

The research works involved literatures review to collect the major issues and concept in the field of machine learning and software defects, attributes reduction, preprocessing of data sets and etc. While collecting all the above resources, the researcher has used different sources like, various articles, journals, books and papers. As a result reviewing and collecting data from different literatures helped the researcher to get better understanding for the research to be most important.

1.7.2. Thoughtful of the Software Defect Problem Area

The researcher has depend on literature review as a base of the study concerning software problem. At the first level, software defect testing and prediction with different data set inside have been studied .In addition to this indirect discuss and interview has been done with domain expert in machine learning as well as software testing area. software that does not work correctly has contain a defect which leads to low quality in software

.Summarily to minimize and fix e defect applying machine learning technology which technique by developing a better model is preferable things .

1.7.3. Understanding the Data for the Study

Once understanding the problem to be addressed evidently in this study, The researcher has introduced the data used for this study after identifying the problem occurred and found in the domain of the software defect researcher engineering. In many studies researchers were used NASA public data sets, Because of this reason, the researcher was choosing this is, the NASA Promise MDP data sets were collected from many different projects developed by many developers and they are public data sets, not private.

1.7.4. Data Preparation For the Study

It is known that about 80% of the total data engineering effort (Shichaozhang,C.Z.aq, 2003) is on data preparation, this implies that much of the research time usually consumed in this phase. After understanding which data sets are useful for the proposed study, the researcher must converted data found to appropriate form. In machine learning ,there are different types of data format ,from the available format. For this study ARFF(attributes relation file format).Weka and Rapid miner are a very comprehensive open source software implementing tool in machine learning. Rapid miner support XML process file as well as Weka support ARFF file. For this reason ,in the researcher has preferred the weka tool for implementation in this study.

Implementation Tools

WEKA tool has been classification used to build models using the algorithms.

WEKA is chosen because, WEKA: is

- Open source.
- Freely available and are familiar.
- Platform - independent.
- Contains complete collection of data processing and modeling methods.

1.7.5. Training and Building the Model

Most researchers were focusing the number of machine learning techniques for their studies A decision threshold optimization on thirteen datasets were applied in (Ayúe Tosun, Ayúe Bener,

2009), shows NB classifier with optimized threshold performs better over default threshold and false alarms, whereas it decreases when analyzed against the basis of receiver operating characteristic curve (ROC) After Naive Bayes, Bayesian networks are believed to give near to accurate results in defect prediction. (Taghi M. Khoshgoftaar, Kehan Gao, Naeem, 2010) compared software prediction models on the basis of four different combinations of original and sampled data for feature selection and data modeling. Filter Based Techniques like chi-square, two types of Relief, symmetrical uncertainty and classifiers KNN and SVM were used.

In addition to this, the researcher was considering, problem domain like, types of data set, nature of project and uncertainty of data set. Mainly the following machine learning techniques are well for the bug prediction in datasets. There are Decision Tree, Support Vector Machine, and Naive Bayes (Saiqa.A, 2015). For comparative analysis the researcher had used various machine learning algorithms (techniques) that were used to build model and tested those data depends on the training model. The 10-Cross validation fold selected for the experiments. The experiment for each machine learning techniques had applied to the selected data sets. Depending on application Classifiers have been used to generate predictor models and it provided prediction.

1.7.6. Performance Testing for Model

The performance measures that are Accuracy based on precision and recall, confusion matrix and the time it takes. The basic purpose of this analysis is to predict defects, examine with the actual prediction one and to measure the performance of the constructed Model.

1.7.7. Prototype Development

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation. In this investigation an attempt to develop software defect analysis and predictive framework system which uses the selected classification rules generated from Decision Tree. The prototype classifies the test data set into defective or not defective class, based on the provided information to it. Graphic user interface (GUI) was developed using Net Beans IDE 8.0.1. For the purpose of loading data, MySQL server data base is used.

1.8. Significance of the study

In this study the researcher has putted improvement of accuracy and performance as a direct significant and as a result, improving the quality of software data in a system by minimizing defects before software release to testing departments and customers. In other hand time taken and cost for testing become reduced.

At the last, a Defective prediction model for attribute reduction use for software developers and testing department in discovering and fixing at the early stage of software defect.

1.9. Organization of the thesis

This research is organized with five chapters.

Chapter One:- Which includes background of study, statement of problem, and motivation of the study, objective of study, specific research question, scope and limitation and significant of study.

Chapter Two:- Includes literature review which gives a detail overview of the study area, software development life cycle, Software defect and predictor, about machine learning, the concept of feature selection and related research work with software defect prediction in previous.

Chapter Three:- Main about research methodology, understanding about data, software metrics, the proposed framework, attributes selection, feature extraction tool and how to evaluate the algorithm used in this study as well as techniques used for performance evaluation .

Chapter Four:- Which covers description data , dealing experiment and analysis result as well as evaluation of the provided machine learning techniques depending on different performance attributes selection to extracts rule for developing of user interface model as the objective of the study.

Chapter Five:-Conclude the research by looking the experiment result with the objective that are proposing before and Provides summary of findings, research limitation, conclusion and recommendation for future work

CHAPTER TWO

LITERATURE REVIEW

2.1. Overview

In this chapter the researcher has generated the theoretical background of the study area and what had been done before and what are the gaps currently there in the area as a core goal of the section. Low software quality may state through several software defects, or software testing and software maintenance may costly due to many defects requiring wide-ranging effort to correct. In this part at the first level, the researcher considered significance of software, software development process, software defect and reasons for the occurrence of those defects. Next, the researcher studied machine learning concepts, classification as well as related fields to the machine learning and the concept of feature selection. Lastly, the researcher discovered the application area of machine learning and related works for software defect prediction in general.

2.2. Software development process (SDP)

Software development process (SDP) or Life Cycle describes the phases of Software Development Life Cycle (SDPLC) and the order in which those phases are going on. It also called a steps take place in any software development process. SDLC compose of various phases or steps which are requirement analysis, System design, implementation, testing and maintenance and (M.Saini and K.Kaur, 2014) (Richard, 2014).

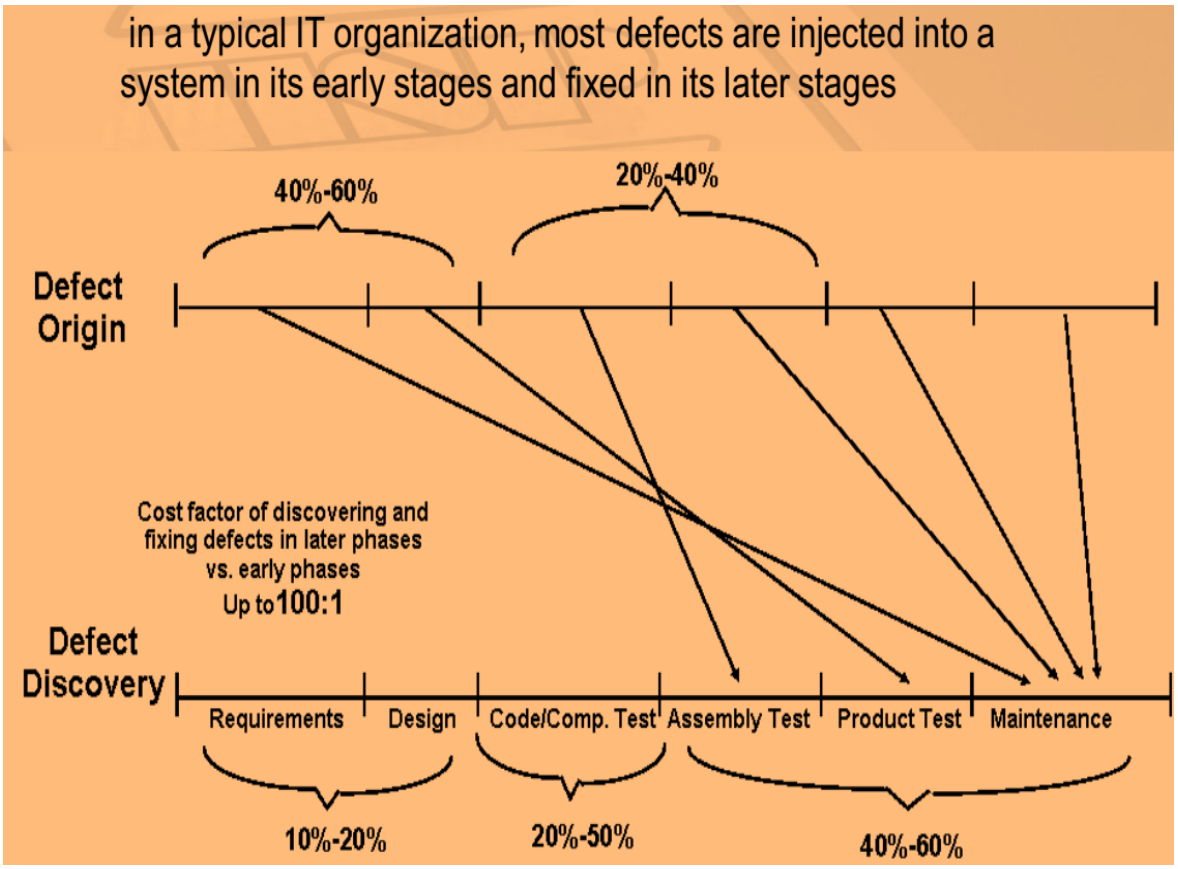

Figure 1:- 2.1 International Institute for software process (IISP), 1996-2008

2.2.1. Requirement Analysis

The researcher has been going to study the feasibility of the software, such as technical, economical and operational, before gathering and analyzing the software requirements. since its a basic in SDLC .The gathered data for system development from different customers would be in this phase. Then the outcome of this would be directly transferred to system design. Meetings with managers, stake holders and users are held in order to determine the requirements. Finally, the software requirement specification (SRS) is prepared which shows that the completion of the first SDLC phase and become an input to the system design phase (M.Saini and K.Kaur, 2014).

2.2.2. System Design

System design is produced from the result of requirement analysis, which are the details on how the system will work is produced. In this phase the project developer determines and understood they are moving in the right track while on the stake holder side, it makes to understand what system design look like for (Richard, 2014).

2.2.3. Implementation

Code is a small unit when software developer produced software for the different purpose of usage. So, implementing test on a small of unit which is code is very important to find and fixing software defect .

2.2.4. Testing

Testing is a means to find and correct errors in software development phases. Testing types can be classified as a unit, Integration, functional, system, user acceptance, beta and regression testing (H.Shivkumar, 2012) (Rasneet K and Iqbal S, 2014) the main function of each testing type is used to evaluate the capability ,usability, correctness and validity the system (software) developed (M.Saini and K.Kaur, 2014) (Rasneet K and Iqbal S, 2014**).** After software requirements are gathered from the users and discussed with project managers, developers and customers. The next thing will be system design and implementation; finally, whether the system is really working with what are gathered from customers would be solved in this phase.

2.3. Software Defect

Any flaw or imperfection in software work product or software process work. In other words Software defect is an errors, that are introduced by software developer and stakeholders. The key objective of software defect prediction is to improve the quality, minimized cost and time of software products. In this section the researcher extremely observed the major factor of software failures that lead the software companies to software defect, consume cost and times to test and maintenance after delivered to the stakeholders.

2.3.1. Major Factors of Software Failure

Our daily life activities are almost connected to Software systems as we have seen in the introduction section parts above. There are many software products or applications that support mainly in terms of business domains (Rajender., 2012), such as in marketing, manufacturing, banking, healthcare, insurance, aviation, media or social networking, e-commerce educational institute or any other domains. There is a need of different resources to develop and design software systems. Some of them are finance, human experts in the domain area, and a lot of time, tools and infrastructures. But in this decade, even if many software companies have a lot of experience in designing and developing projects, (according to Table 2.1) Software failures are increasing from time to time, which leads to loss of capital, time and energy consumption. The system may fail due to defect occurred every

software development cycle or customers may not provide the exact requirements; Because ,they do not have enough awareness of information technology projects or political, cultural issues are there. Software defect is an error that is not meet what the requirement gathered and user expectation. That means, the bugs that are occurred in coding or logic that causes a program to malfunction or to produce unexpected results according to (Pooja P and D.A.Phalke, June 2015).If the program were generating malfunction and produce unexpected error finally it leads to failure. The survey participants were also asked about the factors that cause projects to be challenged. The most common known cause of software failures are Lack of user involvement (Rajender., 2012) unclear goals and objectives (Rajkumar G and K.Alagarsamy, 2013) incomplete requirement specification, lack of resources, inadequate project planning and scheduling, Poor communication between Team members and Poor Testing. From the Table 2.1 below, we have been looking that lack of user input and incomplete user requirement specification are the major factors for projects to be successful. The key factors for software failure are depicted in Table 2.1 below.

Table 1:-2.1 Major factor for software failures (Clancy, 2014).

No	***Project challenged factors***	***% of Response***
1	Luck of user input	12.8 %
2	Incomplete Requirement & specification	12.3 %
3	Changing Requirement & specification	11.8 %
4	Luck Of Executive Support	7.5 %
5	Technology In Competence	7.0 %
6	Luck of Resource	6.4 %
7	Un realistic Expectation	5.9 %
8	Un clear Objective	5.3 %
9	Unrealistic time frames	4.3 %
10	New Technology	3.7 %
11	Others	23.0 %

In the year of 2013, The Company called chaos manifesto reported that, after under taking the survey from year 2004 up to 2012 on software projects and categorizing them based on the percentage of failure, success, and challenging projects as presented blow in table.

Table 2:- 2.2 Percentage detail from 2004 – 2012 (Vikas S and Jatinderkumar R, 2014).

	2004	***2006***	***2008***	***2010***	***2012***
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

2.4. Software Defect Predictor

The software defect predictor is the way or a method that helps for software testing in software development life cycles. Software defect is also referred to as a bug, which can be defined as shortage in the software product that causes the software not to perform its task as the programmer and customer needed. There are different descriptions for software defects, according to the Institute of Electrical and Electronics Engineers (IEEE) standard (Mikyeong P and Euyseok.H, 2014) which are:

***Defect* :-** A flaw in a component or system that can cause the component or system to fail to perform its required function.

***Error* :-** A human action that produces an incorrect result.

***Failure* :-** A failure occurs when a faulty piece of code is executed leading to incorrect state that propagates to the program's output.

***Fault* :-** A fault is the manifestation of one or more errors.

Depending on the above software defect description, a defect that are occurring in software product lead malfunctioning and health problems in case of serious and critical software, such as NASA software products for Space sciences.

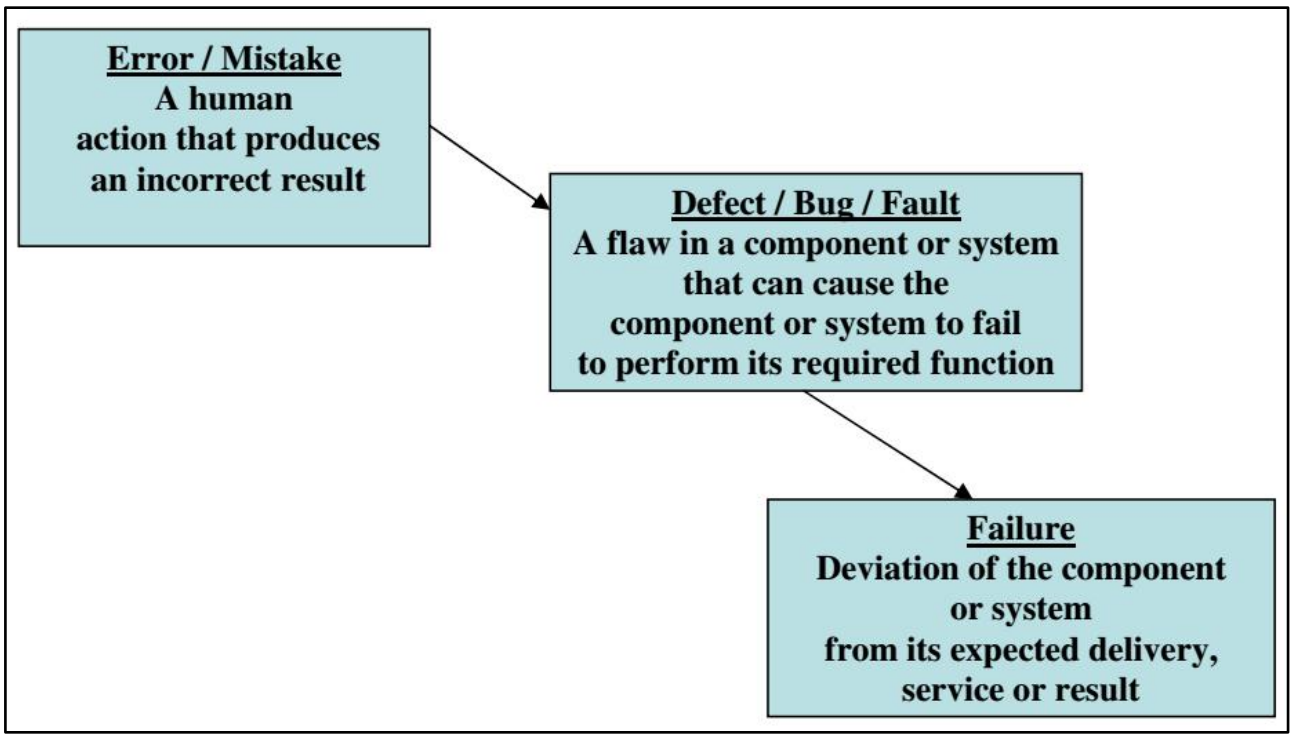

Figure 2:- 2.2 To understand Error, Fault, and Failure (RevaITM)

2.4.1 Software Defect Prediction Process

The process of predicting software defects are very communal to make the use of machine learning techniques that provide computer systems and the ability to learn from data without being unambiguously programmed (Smola,A.,& Vishwanathan,S.V.N, 2008).Initially, data sets are generated from software repositories including defect tracking systems, source code changes and data extraction. Those data sets contain of instances, which can be software components, files, classes, functions or modules. In this study the researcher has depend on the software module component. Based on particular metrics like static code attributes (T.Menzies et al., 2007) extracted modules from the software repositories, an instance is labeled as defective or defect-free.

The collected data sets are then cleaned using preprocessing methods such as noise detection and reduction (Kim et.al, 2011) data normalization (Nam,J.,Pan,S.J.,& Kim,S., 2013) and feature selection (T.Menzies et al., 2007).After that, the preprocessed data sets are used for building a defect prediction model, that is to predict whether new instances contain defects or not. Separately from the

binary classification, this model can estimate the number of defects in each instance. In terms of machine learning, this estimation is also called regression (Smola,A.,& Vishwanathan,S.V.N, 2008).

One of the quality assurance in Software is a defect prediction which activities in Software Quality Engineering such as formal verification, fault tolerance, inspection, and testing. Software metrics (S.Misra, 2011) (O.T.Pusatli,S.Misra, 2011) and fault data (faulty or non-faulty information) belonging to a previous software data instances are used to build the prediction model. The fault prediction process usually includes two consecutive steps: training and prediction. In the training phase, a prediction model is built with previous software metrics (class or method-level metrics) and fault data belonging to each software module.

Software engineering researchers have used Case-based Reasoning, Neural Networks, Fuzzy Logic, Decision Trees, Naive Bayes, Artificial Immune Systems, and several statistical methods to build a tough software fault prediction model. Some researchers have applied different software metrics to build a better prediction model, In addition to this; recent papers (T.Menzies et al., 2007).

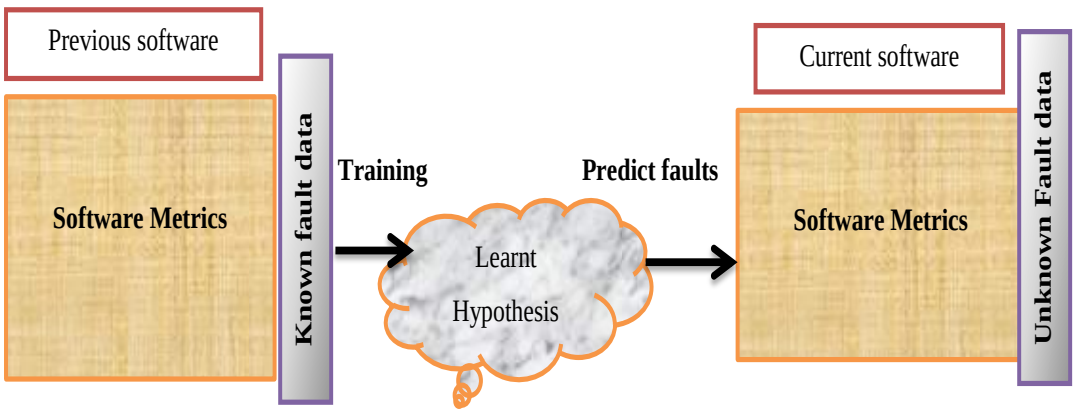

Figure 3:- 2.3 software defect prediction process (Catal, Cagatay, 2012).

2.5. Machine Learning

Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc. Basically, machine learning affords computer programs with capabilities to imitate the human learning process. This process is to observe a phenomenon and to generalize from the observations (Japkowicz,N.,& Shah,M., 2011).In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. The machine learning algorithm can be classified into four, namely supervised, unsupervised, semi-supervised and Reinforcement learning, however the most common known algorithms are supervised and unsupervised learning. There are different machine learning techniques in various research areas, but for this study, the researcher focus on the most mature and widely used one, which is a classification method with an aim of classifying software defects.

2.5.1. Machine Learning Concept

One of the main differences between how people and computer work is that human, performing any kind of activity usually simultaneously expand efforts to improve the way they perform it. This is to say, human can perform any tasks with the versatility means while are not machines. Machine learning algorithms have proven to be of great practical value in a variety of application domains called (Ermiyas, 2016).They are particularly useful for:-

(a) Poorly understood problem domains where little knowledge exists for the humans to develop effective algorithms.

(b) Domains where there are large databases containing valuable implicit regularities to be discovered; or

(c) Domains where programs must adapt to changing conditions.

Machine learning algorithms ‘learn ‘to predict outputs based on previous examples of relationships between input and outputs data’s. The models that are constructed based on the relationship between input and output slowly improved by testing its predictions and fixing when erroneous (A., Harry, 2014); according to Tom Mitchell definition, the machine learns with respect to particular tasks **T**, performance **P** and Experience **E** (Tom M. Mitchell, 1997).The machine learning algorithm can be

classified into four, as listed in section 2.5 above. From the categories, supervised learning requires a training data set with determined labeled class while unsupervised do not required to have labeled class. A problem solved through classification and regressions methods are classified under supervised learning, because of data sets have known output values. Clustering problems can be solved with an unsupervised approach. From machine learning techniques the researcher has explained some of supervised and unsupervised learning shortly in the next section below.

2.5.2. Supervised Learning

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. The outputs can be real numbers in regression or class labels in classification. As classifying an input into two or more classes, supervised learning is sometimes known as classification. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. There are a variety of classification techniques that have been widely exploited in the literature for labeling software instances as defective or defect-free. It is the machine learning task of inferring a function from labeled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value. It has two known supervised learning tasks, classification and regression. Classification concerns on building predictive model for function with discrete range, while regression concern on continuous range model building. A lot of researchers in machine learning were focused on supervised learning (Christian M,Gail K,and Marta A).There are seven most common machine learning tasks that one could come across most frequently while solving an advanced analytics' problems. These are regression, classification, Clustering, Decision Tree, Density estimation, Testing and matching. From the seven most common machines learning the researcher had took only three machine learning techniques for the proposed work in order to use and full field the objective of the paper. These are discussed below here.

2.5.3.Classification

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances)

whose category membership is known. (Jiliang Tang et.al, 2014).Many real-world problems can be modeled as classification problems such as assigning a given email into spam or non-spam" classes, automatically assigning the categories (e.g., "Sports" and "Entertainment") of coming news, and assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).Classification is a widely held method for software defect prediction and categorizes the software code attributes into defective or not defective, which is done by means of a classification model derived from software metrics data of previous development projects. Classification is types of supervised learning where by mapping separate class for instances. The different classes are determined by the output, in ML which known as a label (Kennedy, Credit Scoring Using Machine Learning, 2013) .In classification decision tree is the popular learning and used for approximation of discrete target functions. The decision tree is used to construct a tree structure that represents graphically the training software data. Classification used to classify software faults according to the feature of the item with respect to the predefined set of classes. A general process of software data classification is demonstrated in Figure 2.4, which usually consists of two phases - the training phase and the prediction phase. In the training phase, data set is analyzed into a set of features based on the feature generation models. After representing data set through these extracted features, the learning algorithm utilize the label information as well as the data set itself to learn a map function f (or a classifier) from features to labels as,

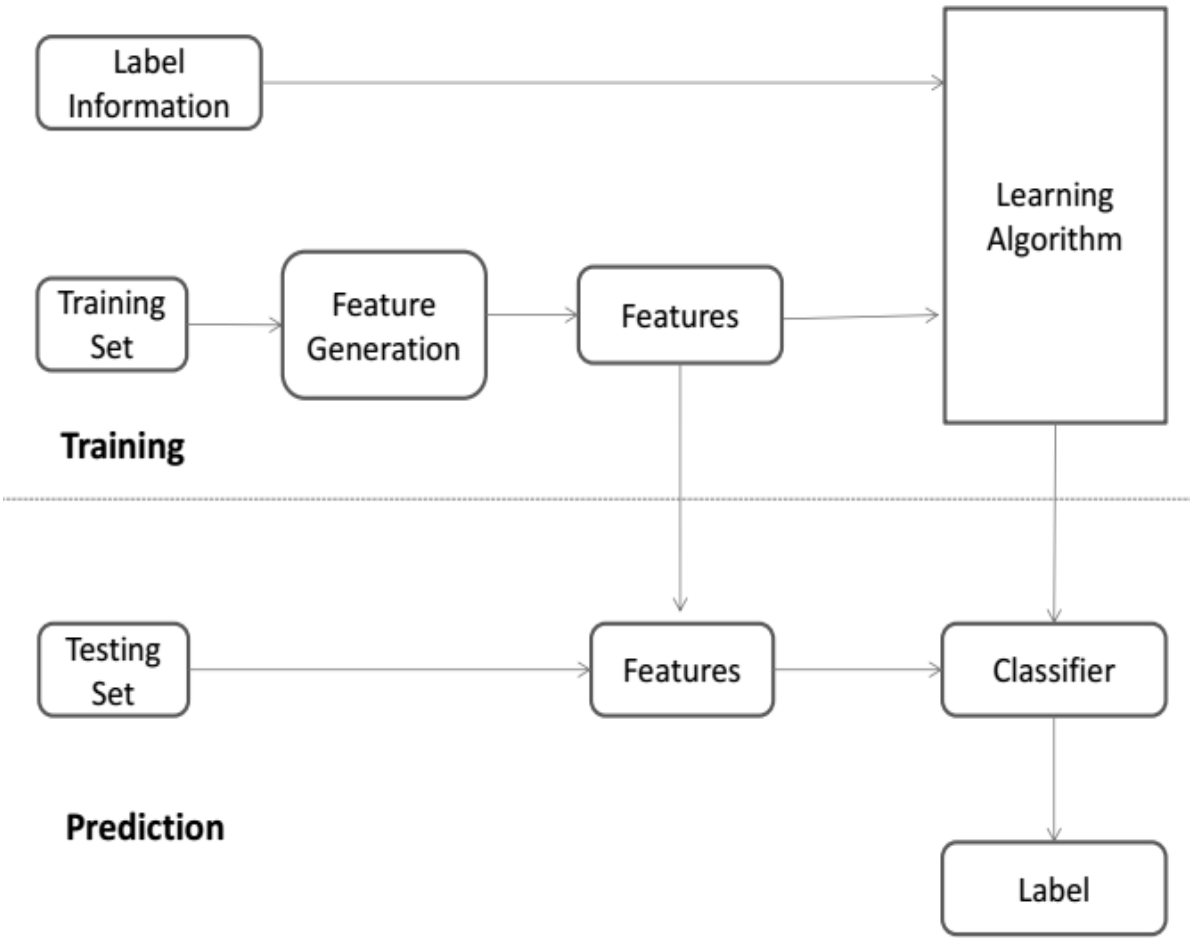

Figure 4:- 2.4 A general process of software data classification.

F (features) → labels……(2.1)

In the prediction phase, data is represented by the feature set extracted in the training process, and then the map function (or the classifier) learned from the training phase will perform on the feature represented data to predict the labels. Note that the feature set used in the training phase should be the same as that in the prediction phase. There are many classification methods in machine learning techniques. Some of the are Naive Bayes (NB) classifier, Support Vector Machines (SVM) classifier, Decision Trees (DT) and Neural Networks (NN) are there. In this study the researcher has selected these three classifiers; Naive Bayes Tree (NBTree) classifier from Decision Tree, Naive Bayes(NB) classifier from Bayes and Logistic classifier from support Vector Machine (SVM). All three learners themselves do not have a built-in feature selection capability and are commonly used in the software engineering community. All classifiers were implemented in the WEKA tool .The researcher used

default parameters as well as reduction parameters settings for the different learners as specified in WEKA.

a) Decision Tree classification

Decision tree is one of the popular prediction algorithms applied to a broad range of tasks in statistics, data mining and machine learning. This algorithm aims to build a decision tree for classifying a target instance based on input features or attributes. It could also be represented as if-then statements for enhancing human readability (Sahana D. C., 2013).Decision tree firstly is constructed by sorting the features down from the root to several leaves. Each leaf node represents a test on a feature while each branch is a possible outcome of the test. In order to label an instance, the tests are taken place at each node from the root to leaf nodes through appropriate branches (Quinlan, 1986).

Table 3:-2.3 Simple decision tree (Sahana D. C., May 2013).

Day	*Outlook*	*Temperature*	*Humudity*	*Wind*	*play golf*
1	**Sunny**	**Hot**	**High**	**Weak**	**Yes**
2	**Sunny**	**Hot**	**High**	**Strong**	**Yes**
3	**Overcast**	**Hot**	**High**	**Weak**	**No**
4	**Rain**	**Cool**	**Normal**	**Weak**	**Yes**
5	**Rain**	**Mild**	**High**	**Weak**	**Yes**
6	**Rain**	**Cool**	**Normal**	**Strong**	**No**
7	**Overcast**	**Cool**	**Normal**	**Weak**	**No**
8	**Sunny**	**Mild**	**High**	**Weak**	**Yes**
9	**Sunny**	**Cool**	**Normal**	**Weak**	**Yes**
10	**Rain**	**Mild**	**Normal**	**Strong**	**No**
11	**Sunny**	**Mild**	**Normal**	**Strong**	**Yes**
12	**Overcast**	**Mild**	**High**	**Strong**	**Yes**
13	**Overcast**	**Hot**	**Normal**	**Weak**	**Yes**
14	**Rain**	**Mild**	**High**	**Strong**	**NO**

In addition to the above table when it expressed by the tree form in the below figure it looks like this.

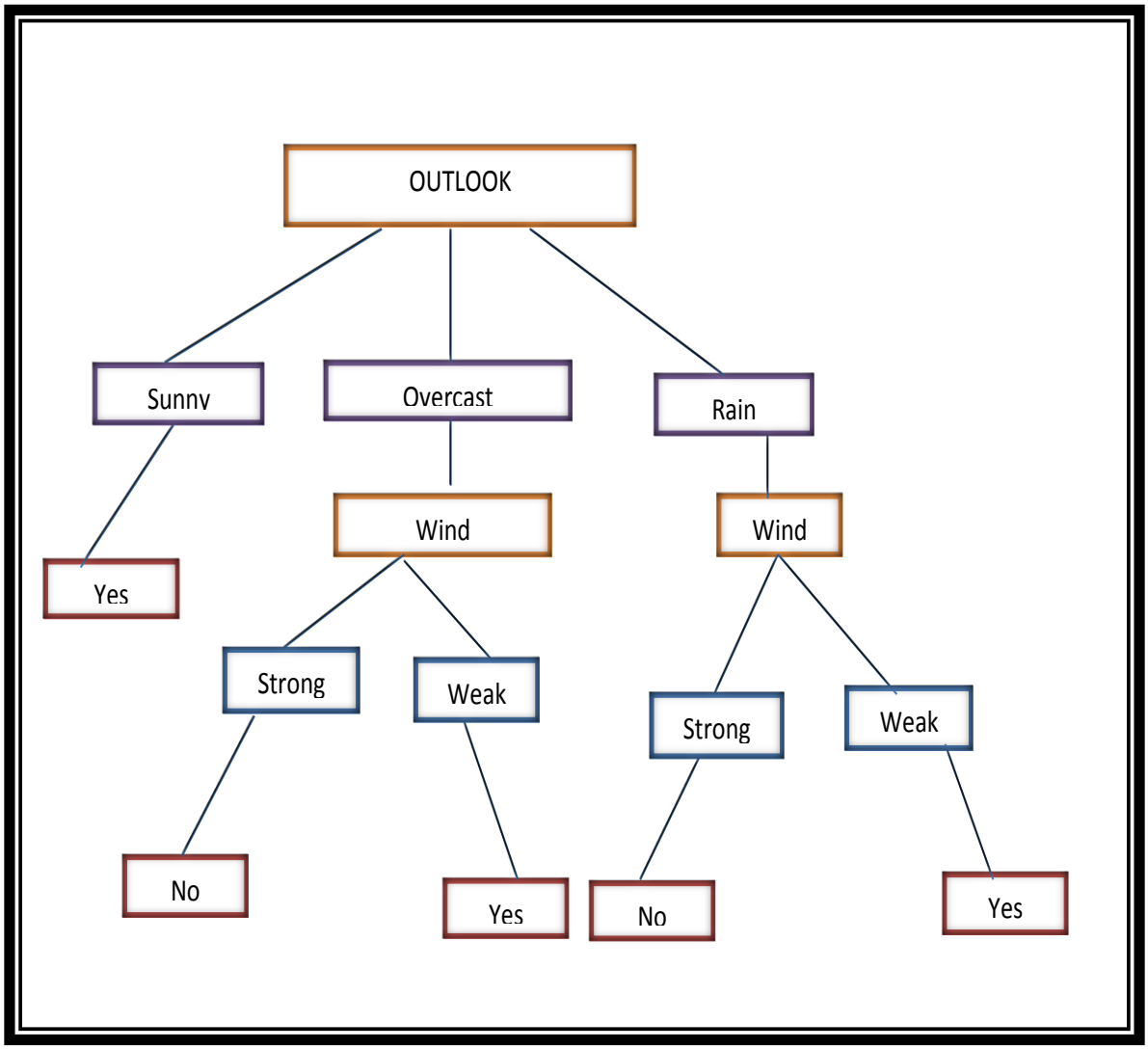

Figure 4:- 2.4 A Simple decision tree (Sahana D. C., 2013).

b) Support Vector Machines (SVM)

The Support Vector Machine approach was first proposed by (Vapnik, 1995). SVM are based on statistical learning theory and neural networks. SVM are very popular in classification and regression tasks and usually present very good prediction accuracy. (Bharat Deshmukh,Ajay S.Patil & B.V.Pawar, 2011) Had used the data sets available from Depaul University and UCI machinery website and From WEKA GUI they have tested the AD Tree (ADT), Bayes Network (Bayes Net),

Decision Table (DT), J48, Logistic, Naive Bayes (NB), NBTree (NBT), PART, RBF Network (RBFN) and SMO algorithms on five data sets, such as bank, car, breast cancer, credit-g and diabetes datasets and observed the following results.

c) Naive Bayes (NB)

The Naive Bayesian classifier is based on Bayes theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

2.5.4. Usage Area of Machine Learning

The aim of machine learning is to address large, difficult and problems on most essential information in a potentially overpower quantitative of data has become increasingly important. Machine learning methods (Algorithm) have practical implications for various application domains; such as poorly understood problem domains, domains where there are large databases and domains that were adapted to change (Zhang) Some of the application area that is addressed by Machine learning are in Web search, spam filters (Ekbal R et.al, 2012) recommender systems, Natural language processing (NLP), credit scoring (Kennedy, Kenneth, 2013) fraud detection, stock trading, drug design, Robot locomotion and many other applications are there. In this study the researcher has given emphasize for the application of machine learning for software defect prediction in software engineering, through a model development for software defect prediction: using feature reduction method.

2.5.5. Machine Learning in Software Engineering

Machine learns automatically from training data and it generate summaries of data or existing systems in a smaller form while software engineer could able to use machines in order to minimize the system development phases time and cost consumptions. One thing that overcomes machine learning integrating with software engineering is developing machine learning based solution to software engineering problems (George T,Ioannis K,Ioannis P,and Ioannis V, 2006).The data and the complexity of pattern in software engineering needs pre-processed before applying machine learning method, this is similar with other applications. Component-based approach fault prevention methods have tremendous attentions from developers now a time, which reduce the development cost and time

as well as to improve the quality and reliability of the projects by breaking the project in to separate components, however it is only practical for finding the problems in the quality of individual components (Xia C,Michael R,Kam-Fai W,and Mabel W). In order to estimate the quality of software we used an attribute such as; software development effort, software reliability and productivity of programmer to predict the important of model in software engineering. Early software quality prediction to enhance the performance of system via machine learning techniques (case based reasoning and fuzzy logic) was studied by (Kennedy, Kenneth, 2013). According to these researches, Software engineering problems that are ready to resolved by machine learning, which are software measurement selection, defect prediction models, software reuse qualification, software requirements gathering, software quality estimation, project management, software testing (A.Harry, 2015) are also called an application of machine learning in software engineering. Among software engineering problems that were listed above, to conduct this study researcher has chosen software defect prediction via machine learning methods. In Table 2.4 above we had seen machine learning approaches (methods) for software engineering tasks.

2.5.6. Machine Learning in Software Defect Prediction

The field of machine learning has been growing rapidly, producing a variety of learning algorithms for different applications. Currently, machine learning is useful for software domain to classify the software modules as defective or not defective, so that early identification of defective modules can be corrected and tested before the final release for the module. As the discussed in section 2.5.3. The ultimate value of ML algorithms are to a great extent judged by their success in solving real-world problems. Therefore, algorithms reproduction and application to new tasks are crucial to the progress of the field. However, various machine learning researchers currently publish for software defect prediction model development. Now, the researcher discussed on different papers done before on the area of software defect prediction by elaborating their aim to solve, methodologies/approaches/ techniques they used, data and their end results accordingly.

2.6. The concept of feature selection in machine learning

As Wikipedia’s definition in machine learning and statistics, feature selection is also known as variable selection, attribute selection or variable subset selection, which is the process of selecting a subset of relevant features, variables and predictors for use in model construction. Feature selection techniques are used for four reasons:-

- Simplification of models to take them easier to interpret by researchers or uses (Gareth James,et.al, 2013)
- Shorter training times,
- To avoid the curse of the dimensionality.
- Enhanced generalization by reducing over fitting (Bermingham,et.al, 2015), formally reduction of variance. (Gareth James,et.al, 2013).

The central premise, when using a feature selection technique, the data contains many features are either redundant or irrelevant, and can thus be removed without incurring much loss of information (Bermingham,et.al, 2015). Redundant or irrelevant features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated. Feature selection techniques should be distinguished from feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points).

Feature selection is one effective way to identify relevant features for dimensionality reduction. However, the advantages of feature selection come with extra effort of trying to get an optimal subset that will be a true representation of the original dataset. Filter methods are feature ranking techniques that evaluate the relevance of features by looking at the intrinsic properties of the data independent of the classification algorithm. A suitable ranking criterion is used to score the variables and a threshold is used to remove the variable below the threshold.

There are many methods to perform Dimension reduction. The researcher has listed the most common methods below. The feature selection methods are typically presented in three classes based on how they combine the selection algorithm and the model building. Feature selection techniques can be categorized into Filter methods, wrapper methods, embedded methods, forward Feature selection, backward Feature selection and hybrid methods.

Filter method

Filter type methods select variables regardless of the model. They are based only on general features like the correlation with the variable to predict. Filter methods suppress the least interesting variables. The other variables will be part of a classification or a regression model used to classify or to predict

data. These methods are particularly effective in computation time and robust to over fitting. they are mainly used as a pre- process method most of the time.

Wrapper method

Wrapper methods evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variables**.** The main disadvantages of these methods are: The increasing over fitting risk when the number of observations is in sufficient. The significant computation time when the number of variables is large.

Embedded Methods

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods. Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm).In addition to the above, there are also another feature reduction method such as forward feature selection, backward feature selection and hybrid feature method.

Forward feature selection

The forward feature selection procedure begins by evaluating all feature subsets which consist of only one input attribute.

Backward feature selection

The researcher selected the filters method. Filters type methods select variables regardless of the model and based only on general features like correlations with variables to predict. Filters methods suppress the least interesting variables. The other variables will be part of a classification or a regression model used to classify or to predict data. These methods are particularly effective in computation time and robust to over fitting. (Hammon, 2013) However, filter methods tend to select redundant variables because they do not consider the relationships between variables. The Correlation Feature Selection (CFS) measure evaluates subsets of features.

2.7. Related Works

Many works in literature related to various feature selection techniques and prediction models using data mining and machine learning methods have encouraged the researcher to propose this paper. In this section the researcher looked at different researches so far done, on software defect specially which have a connected with feature selection using different algorithms approaches.

For starting the related works (Romi Satria Wahono and N.Suryana Herman, January,2014) have investigated improving the performance of SDP by reducing noise attributes through the combination of genetic algorithm and bagging techniques through 9 nine data sets such as CM1,KC1,KC3,MC2, PC1,PC3,PC4 and MW1.In this study they used genetic algorithm to come across with feature selection as well as bagging techniques employed to deal with the class imbalance problem. As a result an impressive improvement in defect prediction performance evaluate was remarkable under 10-fold cross validation.

In addition, to the first selected researchers (Guyon, 2003) Introduced variable and feature selection to improve the prediction performance of the predictors by using efficient search methods and then tested on a wide variety of data sets like text filtering and showed limited possibility of making comparison framework is lacking. Regardless of the fact that a number of approaches have been proposed for effective and accurate prediction of software defects, until now most of these have not found widespread applicability of framework. As a consequence (Vashisht,V.,Lal,M.and Sureshchandar,G.S, 2015) had proposed the effective and acceptable framework for software defect prediction across the software development life cycle using neural network algorithm on real data collected from 15 software companies and through their experiment they had gotten 90% accuracy.

Connecting to this accuracy model (Misha Kakkar and Sarika Jain, 2015) had proposed building framework using attributes selection for perfect prediction depending on five data sets and five techniques such as , IBk, KStar, LWL, Random tree and Random forest. In order to analyzed their datasets for further performance comparison they used Accuracy and ROC value. The outcome result showed that the framework had reduced total number of attributes used for each data set on 6-folds on average. They conclude that, LWL performed better than other remain four techniques. On the other hand, when they are again tested CM1, JM1, KC1, KC3 and PC1 with 10 cross validation and percentage split of 66%. As the researcher knowledge searched for this study most of the researchers were 10 fold cross validation for testing their experiment performance accuracy. According to this

(Ermiyas, 2016) has used 10 fold cross validation through five data sets JM1,CM1,KC1,KC2 and KC3 by different combined techniques such as J48, SMO,SVM and the result recorded was 90.82% performance accuracy in software defect prediction.

Many researchers had used NASA metric data repository in software defect prediction techniques .connecting to this (Jobaer Islam Khan,Alim Ul Gias,Saeed Siddik and Saeed Siddik, 2014)had employed on the problem of attributes selection in the context of SDP by proposing a techniques for selecting best set of attribute to improve the Accuracy of SDP. To achieve their goal the had been used simple algorithm to evaluate these datasets CM1,PC1,PC2, PC3,PC4,KC3 and MW1.pd,pf and balance value examined using Bayesian Network (Zhou Xu et.al, 2015) explored the impact of 32 feature selection methods on the defect prediction performance over two version of the NASA data sets (i.e. the noisy and clean data sets) CM1,JM1,KC1,MC1,MC2,PC1,PC2,PC3,PC4 and MW1.Additionaly open source AEEEM dataset they used a state-of- the art double scott-knott test techniques to analyze those methods. Experimental result show that, the effectiveness of these feature selection methods on defect prediction performance varies significantly over the datasets (Jong and Marchioret.al., 2004)introduced methods for feature selection based on support vector machine learning (SVM).Even though, many researchers had explored in this area still there was a bit challenges there, for that matter (N.Gayatri,S.Nickolas,A.V.Reddy, 2010) had proposed a new methods for feature selection based on Decision tree induction process to achieve better performance by using KC1 dataset .they had been analyzed about 18 classification with three feature selection techniques, J48,BFTree,CART,NB…etc. and ROC,MAE and RME respectively. As a result they had conducted ROC achieves better performance.

(Umar, 2013) Study in software testing or development organization, a need for release project wise better defect and prediction models. Predicting the defect in testing project is a big challenge. He understood that companies spend huge amount of money in allocating resource to testing the software system in order to find the defects according to his investigation of the software development life cycle and account 50% of the total cost of development. He had been used 20 past release data points of software project through multiple linear regression models the R-Square value was 0.91 and its standard error was 59%, at the end he got 90.76% precision between predicted and actual. At last he had calibrated his model accuracy 84%

More than anything in addition to the above mentioned all related woks and the researcher of this study motivation described in chapter one under section 1.3 of this paper is mainly to fill the gap the researcher called (Ermiyas, 2016) done on the complexity of SDP by using five data such as CM1,JM1,KC1,KC2 and KC3,from NASA repository using ML techniques such as J48, SVM, and Vote. To extract the attributes he has used different techniques including RMI Calculator and prest depending on line of codes. As a result he had get accuracy **90.82%** from weka tool in KC3 data set and he had putted feature reduction for SDP as a future work.

Table 4:- 2.4 Related works summary

R.no	***Author name (year)***	***Objective of the study***	***Datasets used By Researches***	***Methodology/approaches/ Techniques/tools/ used***	***Key findings***	***Remarks***
1	Romi SatriaWahono, Nana Suryana Herman (2014)	To deal with the class imbalance problem	CM1, KC1, K3, MC2, PC1, P 3, PC4 and MW1	Combination of genetic algorithm and bagging by 10-fold cross validation	-Impressive had improved	
2	Isabella Guyon (2003)	To improve the prediction performance of the predictors.		Efficient search methods -Tested on a wide variety of data sets	Comparison framework is lacking.	
3	Misha Kakkar et.al (2015)	-Building framework using attributes selection	-Five data sets (CM1, JM1, KC1, KC3 and PC1).	-IBk, KStar, LWL, Random tree and Random forest. -10 cross validation and percentage split of 66%. Were used	-LWL was selected as -good performance.	
4	Jabaer Islam Khan et.al (2014)	-To improve Accuracy of SDP.	-CM1, PC1-PC4, KC3 and MW1.	-Bayesian Network		

5	V.Vashisht et al.(2015)	-To improve effectiveness and acceptance of framework	Real data from different software companies	-Neural Network (NN)	- Accuracy 90%	
6	N.Gayatri et.al (2010)	-To achieve better performance on attribute selection	-KC1 data set	-Decision Tree -18 classifiers (J48, BF Tree, CART, NB…etc)	-ROC achieves better performance.	
7	Shaik Nafeez Umar, (2013)	-To wise project better defect and prediction models	-Used 20 past release datasets.	-Multiple linear regression models.	Accuracy 84%	
8	Ermias Berhanu (2016)	Developing model for Software complexity problem testing	CM1, JM1, KC1,KC2 and KC3.	-J48, SVM, Vote -10 fold cross validation	-Accuracy for KC3 was about 90.82%	Gap there is no feature selection done.

CHAPTER THREE

REASERCH METHODOLOGY

3.1. Software metrics and the proposed approach

Software metric is a quantitative measure of the software; it can be processed metrics and product software metrics. The goal of this chapter is to asses and analyses of different types of software classifiers as well as software metrics used to measure the software product and process in defect prediction by reduction of features (attribute selection). Additionally, description of distinct classification techniques:- Decision Tree (NB Tree), Bayes (Naive Bayes), Support Vector Machines (logistic).Furthermore, after constructing the model, the accuracy and performance of the model is evaluated and the performances of distinct classifiers are compared with each other. Finally, software metrics, the algorithms used to build the models and metrics used for performance measure for a proposed model approach based on Accuracy, precision, recall, and testing options used, 10 fold cross validation are discussed in this chapter.

3.1.1. Framework for software defect prediction model

In this study, firstly the researcher attempts to explore better features reduction techniques for NASA MDP data sets, CM1, KC1, KC2, and PC1. Figure 3.1 below, shows the step by step procedure followed in designing the proposed framework for software defect predication classification. In the proposed framework initially the researchers took data sets from NASA Promise MDP and it has training data which includes software metrics and correspondence values. Secondly, preprocessing of the input data was performed through noise removal and handling missing values. Thirdly, the researchers choose attribute reduction technique in order to select the suitable attributed depend on the output performance to build the model. Fourthly, the researchers choose the algorithms (Logistic, SVM, and Naive Bayes) with 10-fold cross validation test mode. Finally, software defect predictions frameworks are aligned to defected or non-defected one and performance report was explored. In order to assure the design and develop model the researcher develop the prototype system that have an input form automatically feed from database when the user click on load data button to determine whether it defect and non-defect that expressed by the rule extracted.

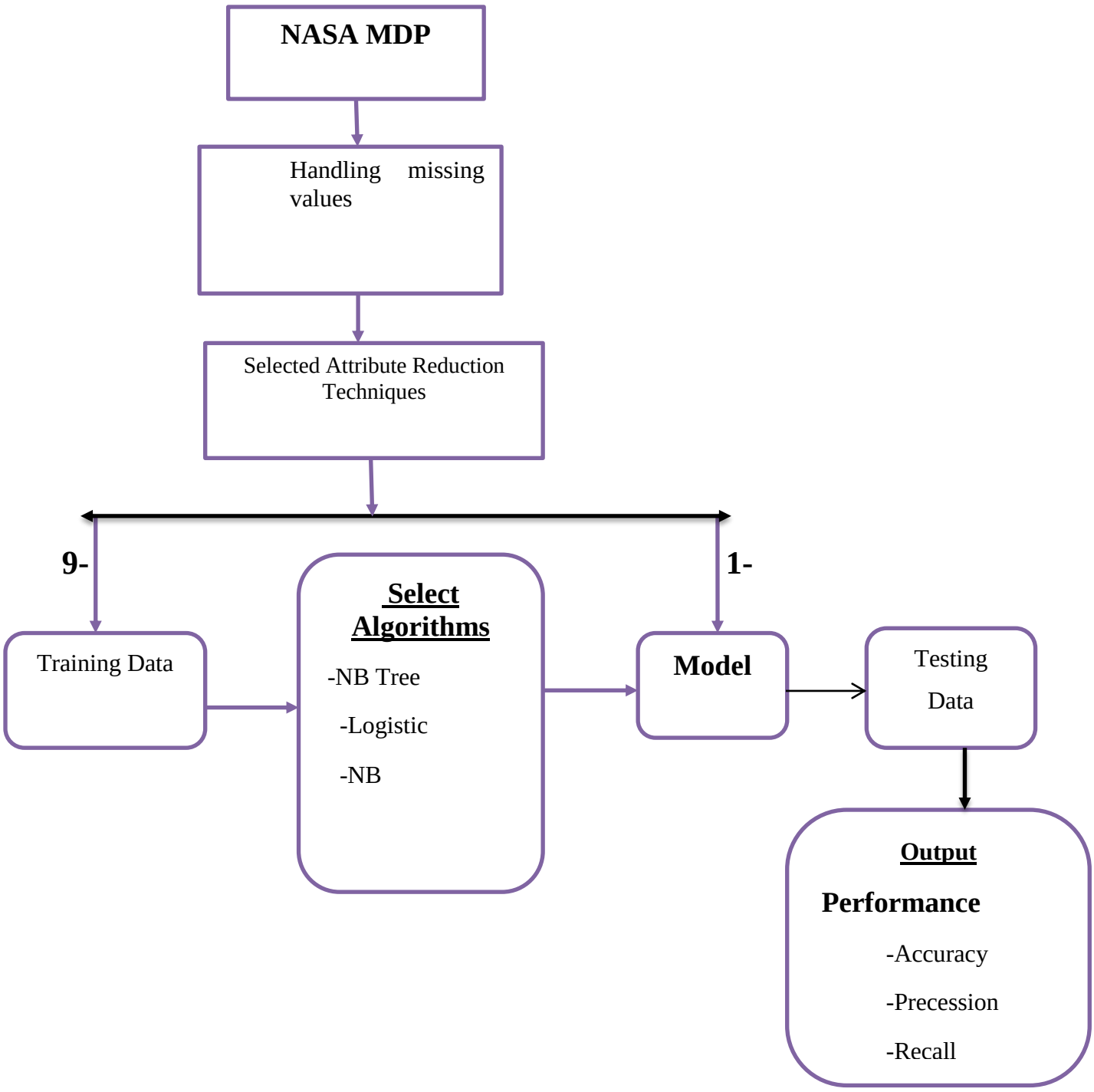

Figure 5:-3.1 Step by step framework for Software Defect Prediction Model.

3.1.2. Software Metrics

Software metrics can be measured as a numerical measurement that allocates symbols or numbers to features of predicted occurrences (Bieman, 1997). In fact, software metrics are features or attributes, which describe many properties such as reliability, effort, complexity and quality of software products. Even for building an effective software defect predictive model. Software metrics play a great role to enhance the software qualities by measuring with product resources and expressed through numerically. Software metrics occurs in the file-level, class-level, module-level, method-level, process-level and quantitative values-level metrics (Malkit S and Dalwinder S, 2013).This helps the project manager to find software defects and making the prevention method for the defect before failure. In this study the datasets used by the researcher had been used measured by module level based metrics, which includes both method and class.

3.1.3. Importance of Software Metrics

Software development phases needs to apply software metrics at each phases. when we see some of them, During requirement analysis software metrics can be developed, for instance, in order to determine cost estimation and resource needed. At the time of system design, we develop metrics in order to count function point. Metrics applied at implementation phase are also used to measure software size (Bhatti). According to (Vikas Vand Sona M, 2011) software metrics have a number of benefits such as provide a basis for estimation and facilitates planning, a means for controlling status reporting, identifying risk areas and effectiveness and efficiency of testing and measuring the software project that has a number of benefits for company it saves development (Vikas V and Sona M, 2011) effort, time and money. The only drawbacks of software metrics is; to better understanding (to acquire knowledge) it needs a lot of effort and time. Software metrics enable software developers or testers to analyze their code and make improvements if required.

3.1.4. Classification of Software Metrics

In order to measure the software from the requirement up to generating of source codes the software metrics can be grouped as follows:-

- Product metrics (code metrics)
- Process metrics
- Resource metrics

3.1.4.1. Product metrics (Code Metrics)

This is one of the software metrics we are applied to measure the quality of the Software systems; mostly it measures the system final product such as; software code or Design documentation. Static code attributes introduced by (McCabe, 1976) and (Halstead, 1977) It can be size metrics, the complexity metrics (Cyclomatic and Halsted). It can also be internal or external attribute measurement of the products (Bhatti). There are many approaches utilizing McCabe and Halstead attributes in combination within machine learning techniques such as; decision trees (J48), k-nearest neighbor, support vector machine and Naive Bayes (Witten, I. H., & Frank, E., 2005) to test software defect. Throughout the history of software engineering, various code metrics have been used for software defect prediction.

a) Size Metrics

Size metric was introduced by Akiyama for the first time (1971). In order to predict the number of bugs, many authors were used number of lines of code as the only metric. The size metrics are try to quantify the 'Size' of software, and the widely used metrics in a Line of Code (LOC). The size metrics are used to measure "volume, length, quantity, and overall magnitude of software products" (Fernando et.al, 2014). The main strength of line of code metrics is easy to count extensively automated for counting and requires various software estimating tools (S.D.Conte et.al,1987).

b) Cyclomatic Complexity Metrics

The word Cyclomatic comes from a number of fundamental cycles in connected, undirected graphs (Arthur H and Thomas J, September,1996).This was proposed by McCabe for measuring the software especially for Design phase. McCabe described by viewing the program graph, and finding the number of paths to reach entire graph. If the complexity of the program becomes long, it is difficult to easily count the number of paths. Due to the above reason McCabe suggests counting the number of basic paths, which is called Cyclomatic number (H.B.Klasky, 2003).Cyclomatic complexity has two basic benefits which giving it the number of recommended tests for software and it is used in all phases of software development cycles, beginning for the design phase in order to achieve software quality parameter(Arthur H and Thomas J, September,1996).Cyclomatic complexity can be expressed as the following equation 3.1.

$$V(G)=E-N+P \quad \text{.....Equation } (3.1)$$

Where:

V (G) = Cyclomatic Complexity

E =Number of edges

N=Number of nodes

P =Number of connected components or parts..

C. Halstead Metrics

Halstead attributes are selected based on the reading complexity of source code. Halsted suggested that measuring the software program by counting the number of operations and operands in the program.

Table 5:-3.1 Halsted Metrics (Menzies et al.,2003)

Name	***Description***
Length:N_1+N_2	The program length
Vocabulary: $\mu = \mu_1+\mu_2$	The vocabulary size
Volume :$V=N_*\log_2 \mu_2$	The information content of a program
Potential Volume :$V^*=(2+ \mu_{2*})\log_2(2+ \mu_{2*})$	The volume of the minimal size implementation of a program
Level: L=V*/V	the program level
Difficulty=1/L	The difficulty level of a program
Error estimate : L	Error estimate for a program
Content I=L*V	The intelligence content of a program
Effort: E=V/L=$\mu 1 N 2 N \log 2\ \mu 2/2\ \mu 2$	The effort required to generate a program
Programming Time=E/18(Seconds)	The programing time required for a program.

3.1.4.2. Process Metrics

This is software metric that is used to measure the quality of the software system. It measures the software development life cycles such as type of methodology, the staff status and the required time to finalize the system. This is mainly measured in software development life cycles and it measures the parameter such as; duration estimate, cost assessment, effort required, process quality and effectiveness or efficiency of the development process (Bhatti). In addition to the above code metrics, the history of software defect prediction has also witnessed the appearance of process metrics. Like code metrics, process metrics are also widely used for building defect prediction models (D'Ambros et.al, 2012).However, rather than directly computed from the existing source code, the process metrics are generated from software repositories such as defect tracking systems and version control systems.

3.1.4.3. Resource Metrics

It is mostly an important measure for resource estimation in software project under project managers. This includes man-power (developer, designer and analyzer), physical resources (computer, material, and methods).

3.2. Object Oriented Metrics

Since object oriented programming becomes popular now a time; as it increases its popularity most companies are going to use it, the complexity and fault prone become a problem so that there is software metrics to resolve this problem before delivered to the customers. The most known object oriented metrics are Chidamber and Kemerer (CK) metrics (R.Chidamber and F.Kemerer, June,1994) Chidamber and Kemerer (CK) metrics were designed to measure the object oriented having the feature such as inheritance, coupling, and cohesion (R.Chidamber and F.Kemerer, June, 1994. In Table 3.1 below object oriented metrics and its description was discussed. Metrics are psychologically attractive and simple from complex OO projects(J.Capers, March ,2006) The drawback OO metrics are do not support studies outside of the OO paradigm, full lifecycle issues, have not yet been applied to testing, maintenance and lack of automation(J.Capers, March ,2006).

Table 6:-3.2 CK metrics and its description

S. No	*Name*	*Descriptions*
1	WMC	Weighted methods per class
2	DIT	Depth of inheritance tree
3	NOC	Number of children
4	CBO	Coupling between object classes
5	RFC	Response for a class
6	LCOM	Lack of cohesion in methods

3.3. Feature Extraction

Every software product needs measurement using software metrics tools. Now a time there is a number of software metrics tools such as Analyst4j ,CCCC, Finder, Eclipse metrics Plug-in, OO-Meter and Prest , however the main difference exist between each metrics are ,license type , platform support and supported metrics. In this study the researcher has used Prest Software Metrics Extraction and Analysis. Prest is exceptional open source and it gathers source code metrics and call graph for five programming languages (C, C++, Java, JSP and PL/SQL). Around 28 basic features could be extracted by Prest from the given source code, the researchers used only 9 attributes from Prest extraction.

3.3.1. Prest Functionalities

In this study as the researcher has discussed the basic purpose of Prest is to extract software attributes from the given source code according to class, method and packages. The important functionality that is supported through Prest is listed below (Ermias Berhanu, 2016).

- Extracting common static code metrics from C, C++, Java, JSP and PL/SQL languages.
- Display the extracted static code thorough graphical user interface in *.xml, *.csv, *.xls, and *. arff file format.
- Generating call graphs in class and method level.

3.4. Attribute Selection or after Preprocessing

Attribute(feature) reduction is the main application method for software data sets in order to reduce its attributes. Here the correlation based feature selection method was used to evaluate subsets of

features. CFS (correlation based feature selection) evaluates a subset by considering the predictive ability of each one of its features individually and also their degree of redundancy (or correlation).In this study when the database was created the domain the researcher selected the attributes that were included on the dataset. As Han and Kamber (2006) stated attribute selection reduces the data set size by removing irrelevant or redundant attributes (or dimensions) and also mining on a reduced set of attributes has an additional benefit. It reduces the number of attributes appearing in the discovered patterns and helping to make the patterns easier to understand.

In order to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes the researcher has used Best First search method. Best First search method starts with an empty set of attributes as the reduced set. The best of the original attributes is determined and added to the reduced set. At each subsequent iteration or step, the best of the remaining original attributes is added to the set. This method selected maximum of 9 attributes from a total of 22 attributes.

3.5. Algorithms Used for Model Construction

There is a number classification algorithm in machine learning and data mining namely, Support vector machine, decision tree, K-nearest neighbor classifier, neural network, Random forest, Naive Bayesians (Tina R and Sherekar S, April 2013). In this section the detail for the algorithms used by the researcher for model construction were discussed.

3.5.1. Decision Trees classification

The Decision tree is one of the most popular approaches for representing the classifiers. Decision tree induction is the learning of decision trees from class-label training tuples. A decision tree is a flow chart like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, each leaf node (or internal node) holds a class label. The top most nodes in a tree are the root node. Internal node is denoted via circles while the leaf node represented by triangles as shown on figure 3.2 Decision tree involves both nominal and numerical attributes. According to Two Crows Corporation (1999) decision tree are the ways that represent the available data in to classes or labels. “Classification trees label records and assign them to the appropriate class**”** (L.Berhe,2011)**.**The researcher has a various reason to selected decision tree which are easy to understand and it is easily converted to a set of production rules.

The decision tree approach is most useful in classification problem. Using this method a tree is constructed for a model through the classification process (Tina R and Sherekar S, April 2013). In order to get a decision, the instance with the highest normalized information gain is used. Then the algorithm recurs on the smaller subsets. The splitting procedure stops if all instances in a subset belong to the same class. Then a leaf node is created in the decision tree informing to choose that class.

There are different decision tree algorithms, such as ID3, C4.5, J48, and NB Tree are some of among the remains one, which can be applied on large amount of instances data and valuable prediction can be produced. Decision tree are preferred because they can evaluate information more accurately than other methods. In this study the researcher selected only one algorithm NB Tree from the decision tree for some reasons. According to (Tang and Zheng al...et., 2005) NBTree is a hybrid approach which includes the capabilities of decision tree and Naive Bayes classifier, the decision-tree nodes contains splits as regular decision-trees, but the leaves contain Naive Bayesian classifiers. Additionally (Gehrke,J.,Ramakrishnan,R.,Ganti,V., 1998) explained NB Tree consists of naive Bayesian classification and decision tree learning. An NB Tree classification sorts the example to a leaf and then assigns a class label by applying a Naive Bayes on that leaf. The steps of NB Tree algorithm are:

- At each leaf node of a tree, a Naive Bayes is applied.
- By using Naive Bayes for each leaf node, the instances are classified.
- (As the tree grows, for each leaf a Naive Bayes is constructed.
- This process repeated until no example is left.

Table 7:-3.3 Training set for predicting borrowers by Decision Tree Hunt's algorithm (Quinlan, 1986).

	Binary	***Categorical***	***Continues***	***Class***
Test Data	***Home Owner***	***Marital Status***	***Annual Income***	***Defaulted Borrower***
1	Yes	Single	125k	No
2	No	Married	100k	No
3	No	Single	70k	No
4	Yes	Married	120k	No
5	No	Divorce	95k	Yes
6	No	Married	60k	No
7	Yes	Divorce	220k	No
8	No	Single	85k	Yes
9	No	Married	75k	No
10	No	Single	90k	Yes

This figure is describe the decision tree more than the table above , Even though their concept are the same .

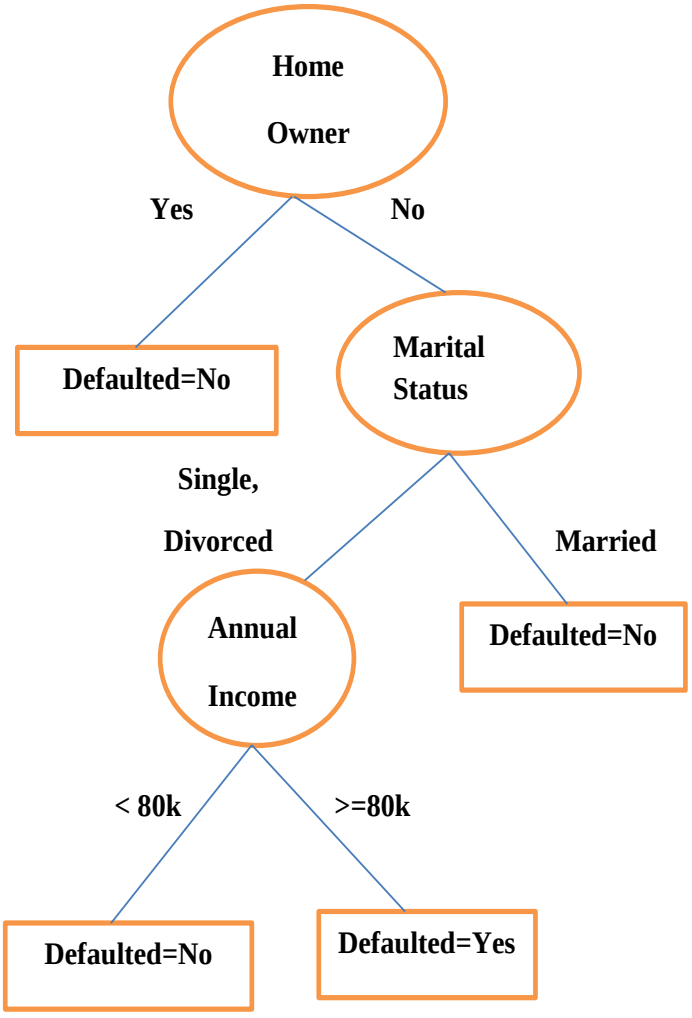

Figure 6:-3.2 Hunt's algorithm for inducing decision tree (Quinlan, 1986).

3.5.2. Bayesian Classification

The Naive Bayesian classifier is based on Bayes theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods. The Naive Bayesian classifier is a straightforward and

frequently used method for supervised learning. It provides a flexible way for dealing with any number of attributes or classes, and is based on probability theory. It is the asymptotically fastest learning algorithm that examines all its training input. It has been demonstrated to perform surprisingly well in a very wide variety of problems in spite of the simplistic nature of the model. It is known that Naive Bayesian classifier (NB) works very well on some domains, and poorly on some.

3.5.3. Support Vector Machine (SVM)

It is a technique for the classification of both linear and non- linear data. Additionally, it is also an algorithm that uses a non-linear mapping to transform the original training data into a higher dimension. The working principles of SVM are based on the concept of decision plane that defines the decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. An appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyper plane. Support vector machine have various characteristics such as ability to handle large feature space, ability to prevent of over fitting and information dense in a given data set (S.Aruna et.al, 2011).SVMs have been successfully applied to a number of real world problems such as face recognition (Edgar O et.al.,1997) text categorization (Susan D et.al), recognition of handwritten digits (O.L.Mangasarian and W.N.Street, 1994),breast cancer diagnosis and prognosis and more.

Logistic is one of the methods used for classification for support vector machine. Logistics is a linear classifier for supervised learning which has properties like feature selection and robustness to noise (Jianing Shi, Wotao Yin et.al, 2010). In this study the researcher has used this algorithm to split the NASA MDP data to fault and non-fault. When running the data using this algorithm we analyzed the classifier output using 10-fold cross validation to make prediction of each instance of the data set.

SVM have been successfully applied to a number of real world problems such as face recognition (Edgar O et.al., 1997).

3.6. Design of Feature Reduction Model for Software Defect Prediction

This research work focused with design and developing model for software defect prediction using feature reduction method testing. First of all the researcher has took the selected data sets from the NASA Promise MDP and it has training data sets which includes software metrics and correspondence values. Second, preprocessing of the input data was performed through noise removal

and handling missing values. Third, applying the chosen methods (NB Tree, logistic and NB), with three different evaluations,10–fold cross validation test mode before attributes reduction as preprocessing parameters as well as after attribute reduction.

Finally, software feature reduction model in defect predictions are aligned to fault or non-fault one and performance report was explored. In order to assure the design and develop model the researcher develop the prototype system that have an input form that automatically feed from data base when the user click on load data button to determine whether it defect and non-defect depending on the selected attributes with rules allowed to display.

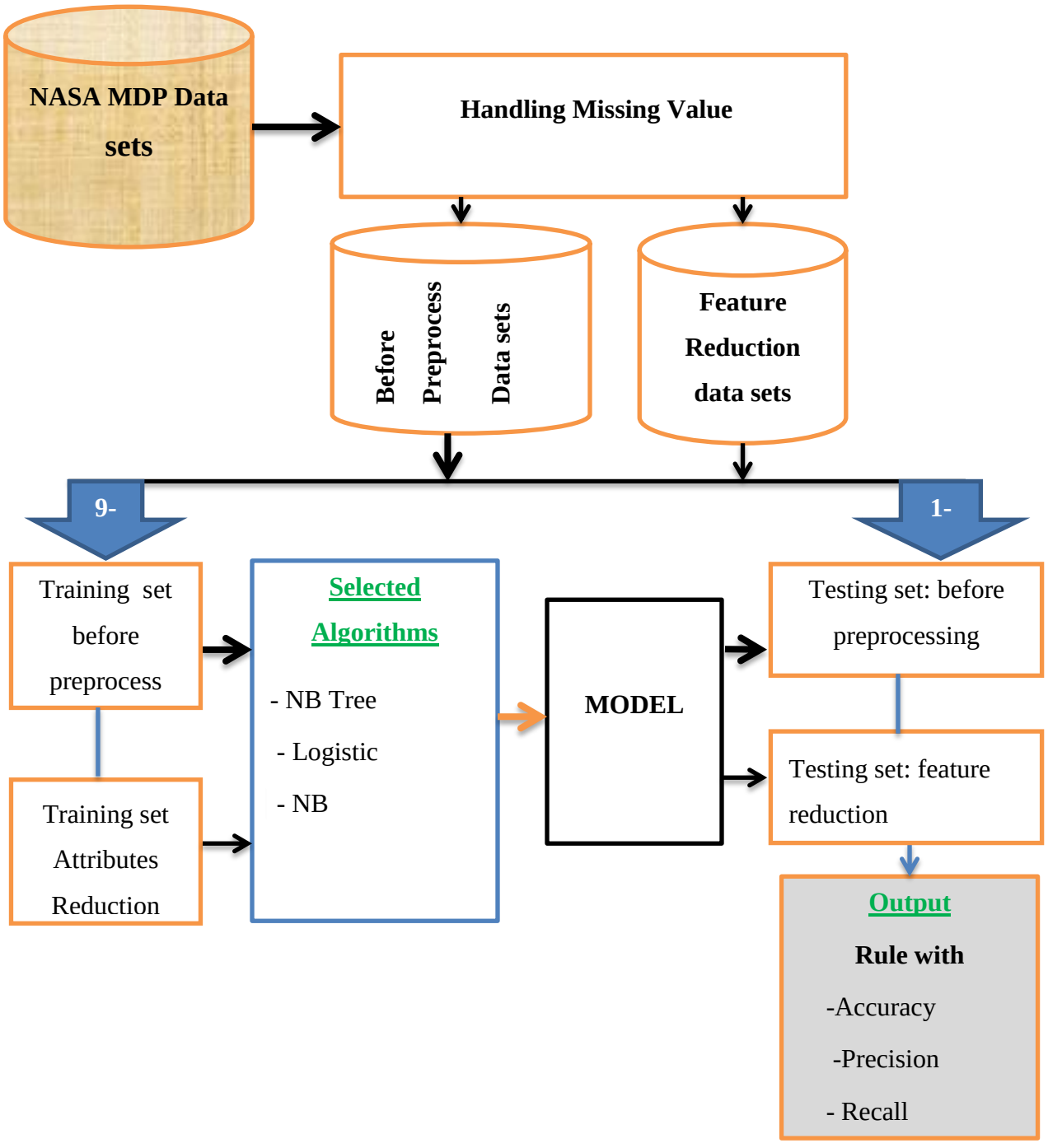

Figure 7:-3.3 A Proposed model for software defect prediction using feature reduction method.

3.7. Performance Measurement

After the researcher has applied the above mentioned different single classifier techniques to the chosen dataset, we should evaluate the performance of the predicated result. In fact, the evaluation of the classifier is an important and mandatory to analysis our result very well. There are numerous performance evaluations in Machine Learning tool (WEKA), including Confusion Matrix, Mean Absolute Error (MAE), F-Measure and Receiver Operating Characteristic (ROC). The most common known performance evaluation is Confusion matrix. The performance of the software defect classification prediction includes error-prone Module detection rate (defect) and Non-error-prone modules misclassification rate (Non defect). In this study the researcher selected numerical (Confusion Matrix and Accuracy based on precision and Recall). Finally, under performance measurement the researcher considered in detail the applied test mode and evaluation method for the research.

3.7.1. K-Cross Validation

K-cross validation is common techniques to estimate the performance of classifier. It is a Key method for assessing an adjustment parameter such as subset. It divide data into K equal parts.one for validation and the remain for training. In this case the researcher has used 10 data for validation or 1 data for testing and 9 nine data for training out of 10 in total which equally divided by machine learning techniques .The researcher has a give set of training examples a single of k-fold validation described below.

3.7.2. Fold Cross Validation

In 10- fold cross validation, is the entire data set randomly dividing into 10- mutually exclusive subset of approximately equal sizes. “The classification model is trained and tested 10 times. Each time it is trained on nine folds and tested on the remaining single fold.” (Damtew, 2011). the performance of the model particularly showed as an average across all experiments The procedure that we followed to perform 10-fold cross validations.

1. The entire dataset is randomly divided into 10 disjoint subsets (i.e., folds) with each containing approximately the same number of records.

2. For each fold, a classifier is constructed using all records except the ones in the current fold. Then the classifier is tested on the current fold to obtain a cross validation estimate of its error rate. The result is recorded.

3. After repeating the step 2 for all 10 folds, the ten cross validation estimates are averaged to provide the aggregated classification accuracy estimate of each model type.

3.7.3. Confusion Matrix

It is also called contingency table or an error matrix. Evaluation of a the performance of a classification model is based on the counts of records correctly and incorrectly predicted by the model. In other hands, Confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true value are known.

Table 3.4 illustrates a confusion matrix for a two class problem having positive and negative class values.

Table 8:-3.4 Confusion Matrix

Prediction class			
Actual Classes		Positive	Negative
	True	True Positive(TP)	False Negative(FN)
	False	False Positive(FP)	True Negative(TN)

From Table 3.3 above the researcher observed that,

TP– stands for the modules are correctly classified as the modules including defects.

FP- means the modules without defects incorrectly classified as the modules including defects.

FN- means the modules with defects incorrectly classified as the modules without defect.

TN- means that the module correctly classified as the module without defect.

Accuracy:- It also called the correct classification rate and used to measure the proportion of the correctly classified modules to the total modules. In this study, accuracy is basic in defect prediction.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \text{.....Equation (3.2)}$$

Even if the accuracy measures the performance of the software defect, it neglects the data distribution character. For instance, when the error-prone module is only a small proportion in the whole module, the classifiers with a high accuracy could classifies the most error- prone module as the non-error – prone module, this shows that it could not satisfy the initial goal of the defect prediction. So that, additional evaluation measure such as, Accuracy, Precision and Recall are needed.

2. Recall:- It the percentage of the correctly predicted positive module in the whole module with defect. If there is higher recall, the less is the error–prone modules not found. Recall can be expressed as follow.

$$\text{Re}call = TP/(TP + FN) \text{.....Equation (3.3)}$$

3. Precision:- It is used to measure the ratio of the correctly classified positive module to the set of positive modules. It is also called true positive rate of consistency. If the percentage of correctly predicted positive module is low or the percentage of incorrectly classified as negative module is high, a low accuracy would measure. Precision can be expressed as follow.

$$Percision = (TN)/(TP + FP) \text{.....Equation (3.4)}$$

4. Specificity:- Usually there is a compromise in the recall and the specificity when the recall is high; it means that the classifier has high.

$$Specificity = (TN)/(TP + FP) \text{.....Equation (3.5)}$$

CHAPTER FOUR

DATA UNDERSTANDING AND EXPERIMENTATION WITH ANALYSIS

In this chapter the researcher was depend on the data collected from NASA MDP (for research determination data set developed by NASA); in order to well understand the data, a close relationship has made with my advisor, my senior researcher on the area of machine learning using NASA datasets and Domain Experts such as software developers and software testers. To summarize, this chapter, the researcher looked at how each data set was understood, prepared for machine learning tools and processed goes through different types of experiments using some selected machine learning techniques and analyzing the results of the experiments by some one of performance comparisons in order to compare with the past researcher results and to preferable result.

4.1. Data Understanding

The key of software defect prediction is how to effectively analyze and use existing historical data for creating more precise classifiers. (Jing and et...al., 2014) NASA MDP is one of open source software which is publically available for many researchers. NASA MDP contains software metrics as an attribute with in data set and show that either a particular data is defective or non-defective. The data set is available in NASA MDP Promise repositories (NASA).For this study, the researcher was collected and used data from NASA MDP Promise due to its original version and 60% of software fault studies were selecting it as a Priority (M.Ruchika, November2015). The researcher has taken into account different things about the data set information's before relating to machine learning techniques, which are discussed in the below 4.1.2 sections.

4.1.1. Choosing and Collecting Data Set

The early steps is to select and collecting the related data from available data sets, that correctly described and encounter what the researcher need in the software fault prediction purpose. Based on NASA Promise repositories (D.Gray,D.Bowes,N.Davey,Y.Sun,and B.Christianson, 2012) currently consists 13 data sets described intended for software metrics researches. Each data sets entitled a NASA software system or subsystem and holds the static code metrics and corresponding fault data for each comprising module (D.Gray,D.Bowes,N.Davey,Y.Sun,and B.Christianson, 2012).To identify more clearly 'module' can mentioned to as a function, procedure or method depending on the programming languages. Measurements are based on line of code, while other compromised

measurements are Halsted and Cyclomatic. According to the scope and limitation of the this paper that was described under chapter one in section 1.6 above, from 13 data sets the researcher has picked only four data sets (CM1, KC1, KC2 and PC1) because of it consisting different programming language, code metrics (Halstead's complexity, code size and McCabe's Cyclomatic complexity) and time constraints.

4.1.2. Description of the Selected Dataset

For this study the researcher has defined dataset as collection of data used for some specific machine learning purpose according to (Sammut,C.,& Webb,G.I., 2011).The selected dataset is explained in detail based on its instance, attributes, programming language and its application as follows:-

- **CM1:**- is a NASA spacecraft instrument written in C language; data come from McCabe and Halstead feature extraction of the source code. These features were defined in the 70s an attempt to objectively characterize code features that are associated with software quality.CM1 has 498 instances and 22 attributes (5 different lines of code measure, 3 McCabe metrics, 4 bases Halstead measures, 8 derived Halstead measures, a branch-count, and 1 goal field).

- **KC1:**- is a C++ system implementing storage management for receiving and processing ground data. It consists of 2109 instances and 22 attributes (5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, a branch-count, and 1 goal field).

- **KC2:**- is Data from C++ functions. Science data processing; another part of the same project as KC1; different personnel than KC1.Shared some third-party software libraries with KC1, but no other software overlap. It includes 522 instances. It also included 22 attributes (5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, a branch-count, and 1 goal field).

- **PC1:-** is Data from C functions flight software for earth orbiting satellite. It consists of 1109 instances and 22 attributes (5 different lines of code measure, 3 McCabe metrics, 4 base Halstead measures, 8 derived Halstead measures, a branch-count, and 1 goal field).

To summarize, the basic information about the data set that the researcher has used through this study is shown in the following Table 4.1 below.

Table 9:-4.1 Data set information

data set	*Language*	*LOC*	*Module*
CM1	C	20k	498
KC1	C++	43k	2109
KC2	C++	18k	522
PC1	C	40k	1109

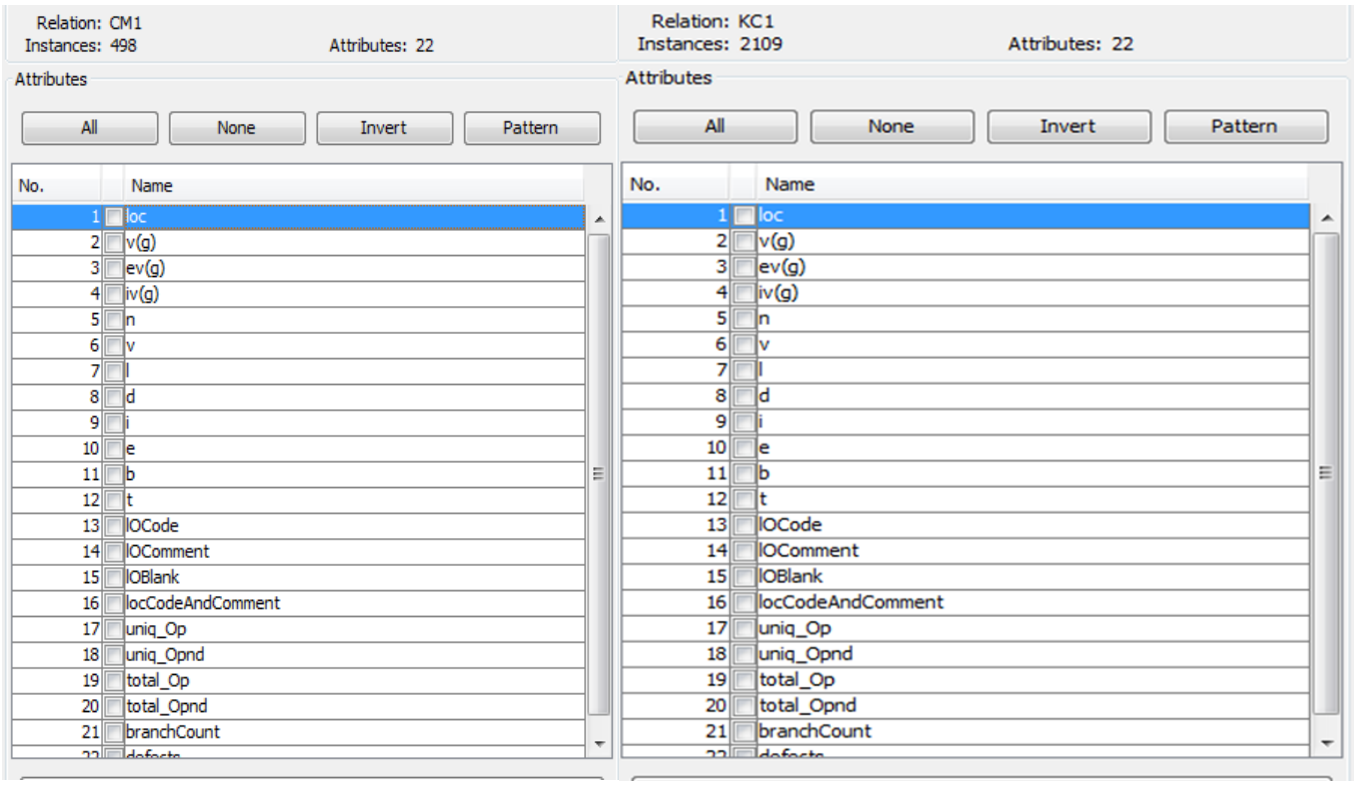

Figure 8:- 4.1 Screen shoot of CM1 and KC1 data set

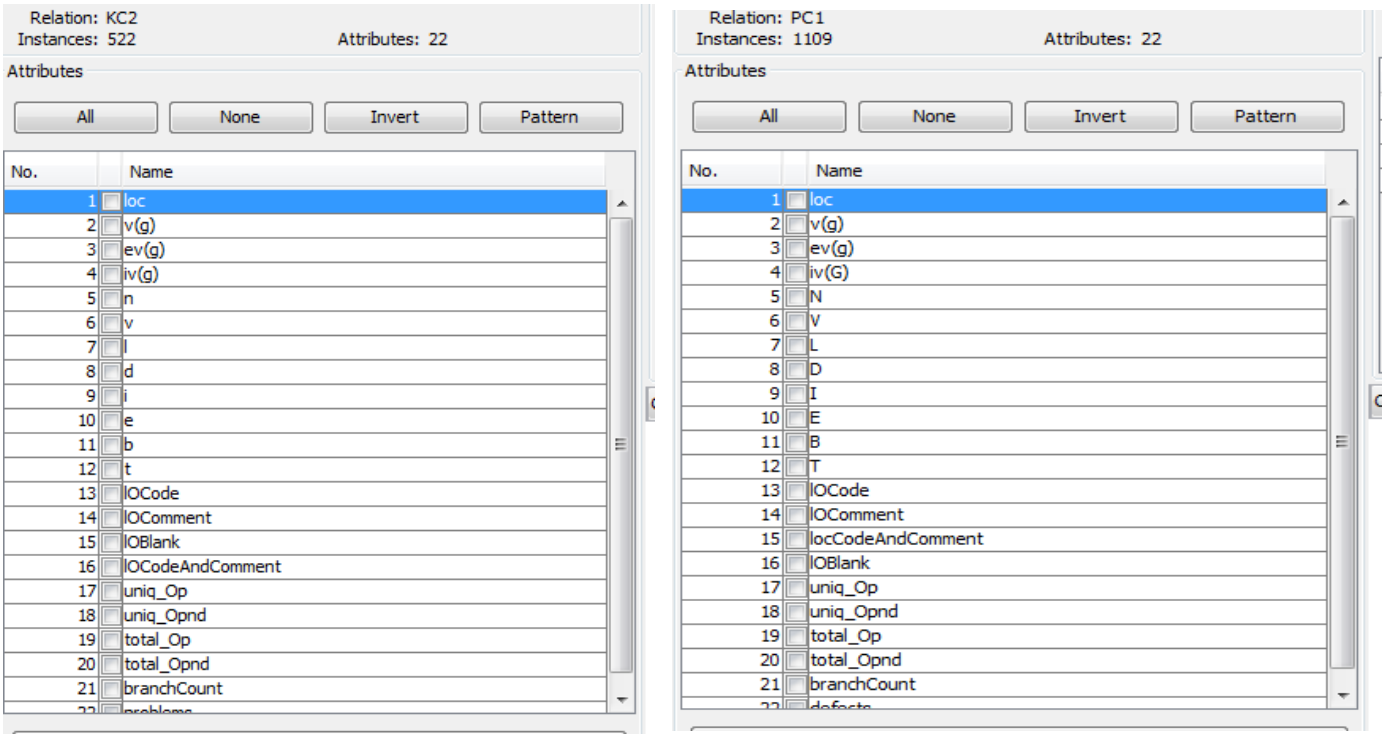

Figure 9:- 4.2 Screen shoot of CM1 and KC1 NASA data set.

4.2. Data Preprocessing

Preprocessing data are good systems, which are very important and widely used in machine learning. The use of data processing is to classify different data properly. When the data set has not been processed well it prone to non-normal distribution, correlation and redundancy of data. Using the unprocessed data affect both the accuracy and the performance of the constructed model, hence many the researchers have gone through various ways of data processing, namely:-Data Cleaning, Correlation analysis (selection) and data integration and Transformation. In this study the researcher has selected some parts of Data cleaning and Correlation analysis only in order to preprocess the data set. Feature extraction was already conducted by software metrics tools for each software product; so that the researcher would apply these techniques' in this study using Prest.

4.2.1. Data Cleaning

The main objectives of cleaning the data were to eliminate the noise data, handling the missing values and handling outliers. In this study the data cleaning conducted for the purpose of making the data set to safe in classifier methods, as follow:

4.2.1.1.Handling Noise Data

Noise data are errors that are occurred during collecting, measuring modules and encoding by human or computer errors. Handling noise data is important to good understood of data imperfection. In order to build and evaluate prediction models, defect data is often extracted from log files, version controls and bug reports in bug tracking databases automatically by using tools (Kim et.al, 2011).The data were filtered before being used in machine learning tool. Instance that are believed of being unclean according to researcher's evaluation criteria's were removed. In order to handle the noise data in this study through filter the instance that had tested. But, In this paper the researcher hasn't get any visible noise data while experiments were running.

4.2.1.2.Handling Missing Values

Any researchers or experimenters should give attention on the handling missing values in the data set. Several literatures suggested that malfunction measurement of equipment and mistakes has occurred during data collection, as well as collection of several similar but not identical datasets. Missing values frequently indicated by out of range entries, in case of numerical attribute values may be negative number (e.g. -1),but it should normally be positive or zero (0) and for nominal attribute the missed value may be shown as blank or dashed. In this study the missing value occur due to partial or incomplete measuring of software product through software metrics tools. The data set is missed values indicted with left blank and replaced 0 for numerical and NULL for nominal attributes using WEKA (Filter→ Unsupervised →Attribute → Replace Missing Values). Replace missing values were applied to all attributes in the given data sets (CM1, KC1, KC2, PC1); However, there is not any missing values recorded for each attributes.

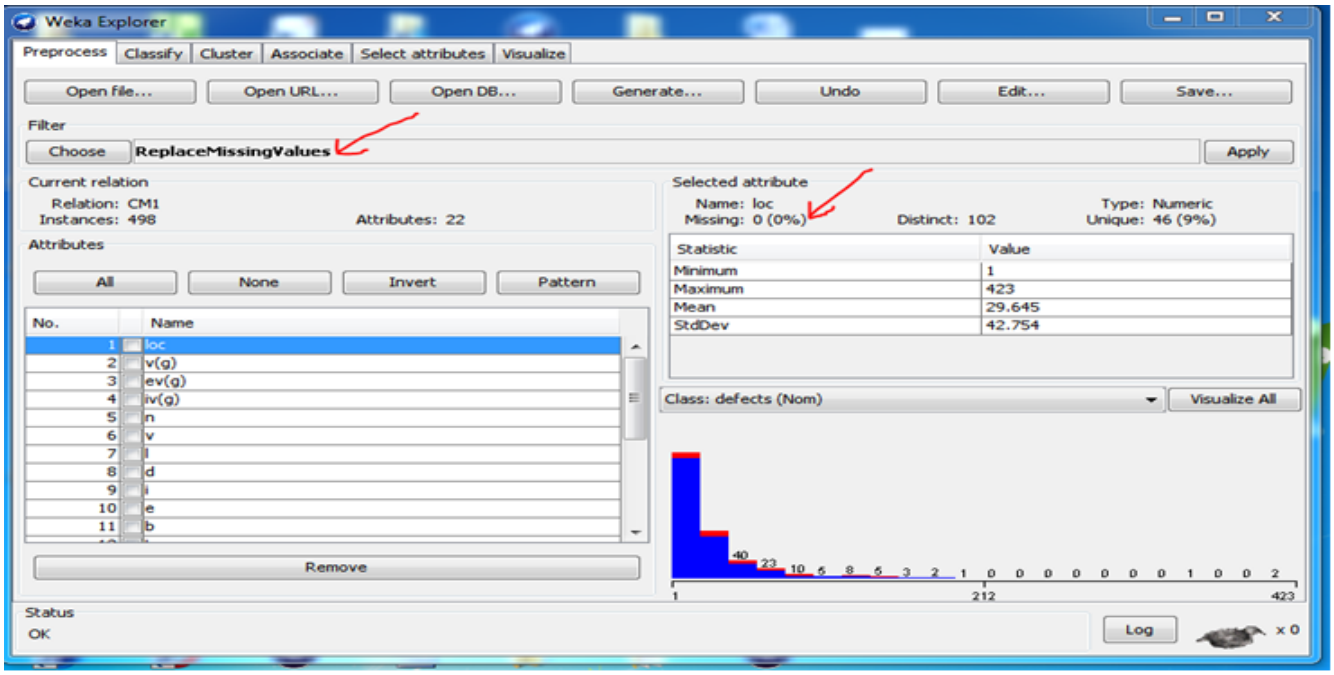

Figure 10:- 4.3 Screen shoot to show missing values.

4.3. Attribute Selection and Discussion

The attribute used for this study is listed on Table 4.2. Now we look in detail how the numeric value of the software product is measured through software metrics tools. As the researcher discussed in chapter three above the software metrics is mainly classified as process and product metrics. However, the target of this study was product metrics. Software products were measured by McCabe, Halsted and Line of Code metrics. The McCabe metrics are a collection of four software metrics namely Essential complexity, Cyclomatic complexity, Design Complexity and line of code (LOC). Cyclomatic complexity [v (G)], is used to measure the number of linearly independent paths. (Bhatti).

Table 10:- 4. 2 Attributes and their description for CM1, KC1, KC2 and PC1 data sets.

No.	*Attributes name*	*Data type*	*Description*
1	Loc	Numeric	McCabe: cyclomatic complexity
2	v(g)	Numeric	McCabe: essential complexity
3	ev(g)	Numeric	McCabe: design complexity
4	iv(g)	Numeric	Halstead: total operands and operators
5	N	Numeric	Halstead: volume
6	V	Numeric	Halstead: program length
7	L	Numeric	Halstead: difficulty
8	D	Numeric	Halstead: intelligence
9	I	Numeric	Halstead: effort
10	E	Numeric	Halstead: delivered defects
11	B	Numeric	Halstead: time estimator
12	T	Numeric	Halstead: line count
13	lOCode	Numeric	Halstead: line count of comments
14	lOComment	Numeric	Halstead: count of blank lines
15	lOBlank	Numeric	Line count of code and comment
16	lOCode And Comment	Numeric	Line of code and comment
17	uniq_Op	Numeric	Unique operators
18	uniq_Opnd	Numeric	Unique operands
19	total_Op	Numeric	Total operators
20	total_Opnd	Numeric	Total operands
21	branch Count	Numeric	of the flow graph
22	Defects	Text	Module has not one or more reported defects

4.3.1. Prest Based Extracting Attributes

The researchers used C Program to find the first capital letter in a string using Recursion function as shown Appendix 2. Prest can Parse all file that are written in C, C++, Java, JSP and SQL thorough different parsers at the same time. One of the project like sample code that the researcher has presented in appendix 2 is parsed and the metrics are presented as figure 4.2 below.

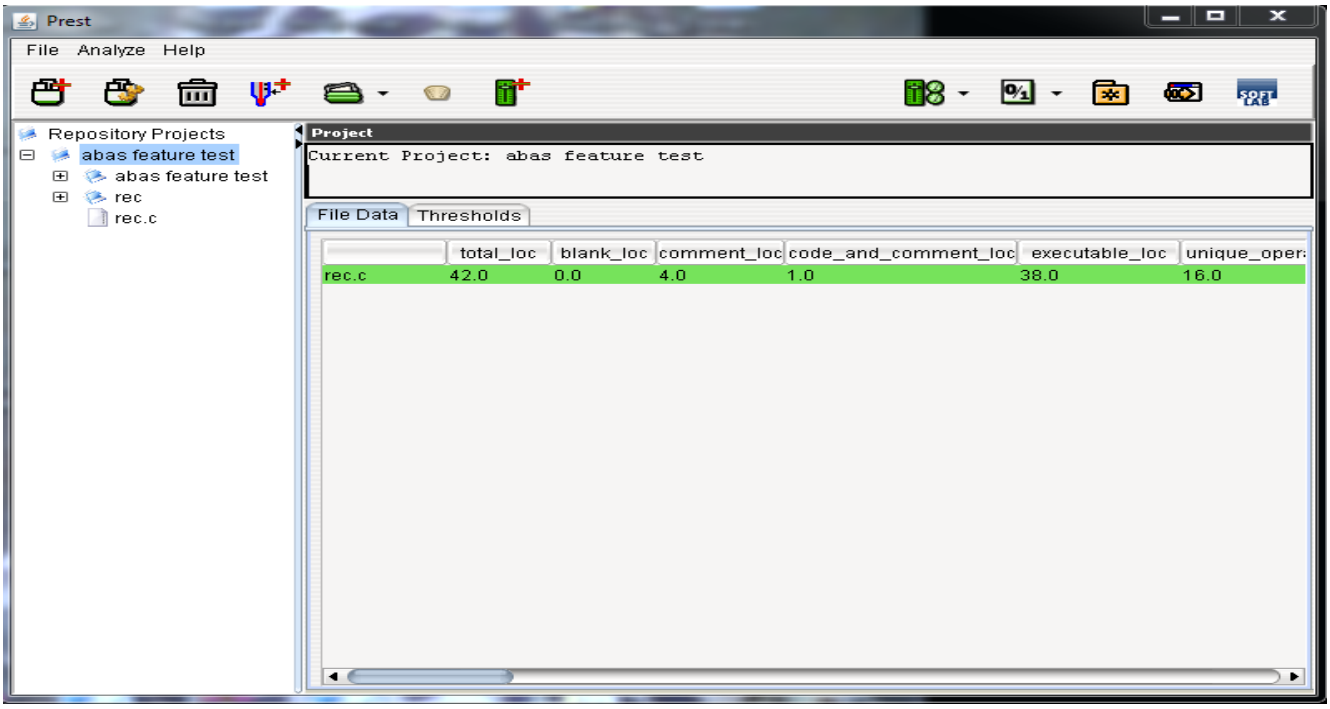

Figure 11:- 4.4 Attribute extraction by prest from file data.

4.4. Experimentation and discussion

As the main purpose of this study is a model development for software defect prediction: using feature reduction method through a machine learning approach by different classifiers, before preprocessing and after attribute selection or feature reduction, then comparing their results according to their performance accuracy, precision, recall as well as time take for processing the selected algorithms and lastly the best algorithm was selected for building a proposed model. The model was built through (NB Tree) from Decision tree, (Logistic) from Support Vector Machine, (Naive Bayes) from Bayes by using WEKA 3.6.9 version of Machine Learning Software Tool. Additionally the experimentation and developed model are analyzed using with specificity based performance measure. The selected evaluation techniques support the researcher to compare the performance of

the chosen classifiers algorithms and make the conclusion depending on result that had found at the end of the experiments.

4.4.1. Experiment setting

In this study, the researcher directed the experiment in two ways categories namely classifier before and after attributes reduction; the researcher has done twenty four experiments, Twelve of them were before preprocessing and the remains twelve's were after attribute selection to develop the preferable predictive model to evaluate each four datasets separately through three machine learning techniques. That means, one data set has tested two times by each techniques and six times by entire techniques.

The experiments were conducted on a full training dataset containing 498, 2109,522 and 1109 instances CM1, KC1, KC2, PC1, respectively and 10-fold cross validation test mode was implemented for randomly sampling the training and test sets. The outputs of models in this study were evaluated using with the standard metrics like accuracy, based on precision, time and recall as mentioned above. In addition to developing and measuring of the performance of models; the researcher also developed the predictive prototype system using Net Beans IDE 8.0.1 in order to assure the predictive model exactly predicting by feeding user input that were taking from software modules after it has been measured via prest software metrics tools.

Table 11:- 4.3 Reduction of attributes on chosen datasets.

R.No	***Selected Data set***	***Instances***	***Attributes***	
			Before preprocessed Nominated	***After Nominated***
1	CM1	498	22	8
2	KC1	2109	22	9
3	KC2	522	22	6
4	PC1	1109	22	7

4.4.2. Model Building using ML Classifiers

The main goal of Under this section the research is to improve the accuracy and performance of selected ML classifiers. In order to do this, first of all the researcher should measure the performance

and accuracy of each single classifiers before preprocessing and after attributes reduction going on ,lastly record it based on the method and performance measure that were selected for the study. The chosen algorithms are three namely; Naive Bayesian Tree (**NBTree)**, Naive Bayesian (**NB)**, and **Logistic**.

4.4.2.1. Naive Bayes Tree Classifier

In this experiment, the performance of NBTree classifier in predicting software defects using four datasets was evaluated. Eight models were built using NB Tree with default parameters and all attributes. Table 4.4 shows the results of experiment1.With CM1 dataset, out of 498 instances, 440 were correctly classified and the remaining 58 (11.64 %) instances were incorrectly classified and, precision and recall of the model was 0.98 and 0.938 respectively. In addition to this CM1 dataset after attributes reduction, out of 498 records 442 were correctly classified and the remaining 56 (11.24%) instances were incorrectly classified and, precision and recall of the model was 0.90 and 0.98 respectively. With KC1 dataset, out of 2109 records, 1796 were correctly classified and the remaining 313 (14.84%) instances were incorrectly classified and, precision and recall of the model was 0.86 and 0.97 respectively. In addition to this KC1 dataset after attributes reduction, out of 2109 records 1793 were correctly classified and the remaining 316 (14.99%) instances were incorrectly classified and, precision and recall of the model was 0.86 and 0.97 respectively. With KC2 data set, out of 522 instances, 431 were correctly classified and the remaining 91(17.56%) instance were incorrectly classified and, precision and recall of the model was 0.85 and 0.94 respectively. In addition to this KC2 dataset after attributes reduction, out of 522 records 438 were correctly classified and the remaining 84 (16.09%) instances were incorrectly classified and, precision and recall of the model was 0.86 and 0.94 respectively. With PC1 data set, out of 1109 instances 1037 were correctly classified and the remaining 72(6.49%) instance were incorrectly classified and, precision and recall of the model was 0.94 and 0.98 respectively. In addition to this PC1 dataset after attributes reduction, out of 1109 records 1037 were correctly classified and the remaining 72(6.49%) instances were incorrectly classified and, precision and recall of the model was 0.94 and 0.98 respectively.

Here the amount of PC1 dataset before preprocessing and after attributes reduction are the same or equal but different attribute as a result it scores high precision, high recall and high accuracy in both before and after attribute reduction.

Experiment 1

Table 12:-4.4 Decision tree result with all before preprocessing and after reduction of attributes.

Models	*Number of Instance*	*CI*	*ICI*	*Number of leaves*	*Size of*	*Time(sec)*	*precision*	*Recall*	*Inaccuracy*	*Accuracy*
CM1,22	498	440	58	3	5	0.42	0.98	0.938	11.64%	88.35%
CM1,8	498	442	56	3	5	0.27	0.90	0.98	11.24%	88.75%
KC1,22	2109	1796	313	5	9	7.82	0.86	0.97	14.84%	85.01%
KC1, 9	2109	1793	316	5	9	1.98	0.86	0.97	14.98%	85.15%
KC2,22	522	431	91	3	5	1.9	0.85	0.94	17.43%	82.56%
KC2, 6	522	438	84	2	3	0.09	0.86	0.94	16.09%	83.90%
PC1,22	1109	1037	72	11	21	4.84	0.94	0.98	6.49%	93.5%
PC1,7	1109	1037	72	11	21	0.31	0.94	0.98	6.49%	**93.51%**
10-fold Cross Validation										

4.4.2.2. Naive Bayes Tree Classifier Evaluation

Eight experiments were conducted based on all attributes and within four datasets to evaluate the performance of NB Tree classifier. According to this datasets, the results of NB Tree classifier in Table 4.4 show that when the number of attributes decreased in preprocessing, the numbers of time taken to build models were decreased whereas Recall, precision and accuracy of the model were increased or equal. Accuracy in NB Tree each and every datasets increases with the reduction of attribute or with attributes selection. Additionally, Accuracy in NB Tree increases in each and every selected datasets, while the time taken to perform the process decreases, when attribute selection taken place. In this experiment the highest accuracy has appeared in PC1 data set as well as the lowest accuracy performed in KC2 data set. In general as the instances in data set increases the time taken to perform has also increases, according to this KC1 data set which has greatest number of instances size has the highest number of time when we compare to the remain three instances.

Table 13:- 4.5 Evaluation of data before preprocessing and after attribute reduction by (NB Tree)

Evaluation of data before preprocessing					*Evaluation of data after Attributes Reduction*				
Data Sets	*Time taken(s)*	*Precision*	*Recall*	*Accuracy*	*Data Sets*	*Time taken(s)*	*precision*	*Recall*	*Accuracy*
CM1	0.42	0.98	0.93	88.35%	CM1	0.27	0.90	0.98	88.755%
KC1	7.82	0.86	0.97	85.01%	KC1	**1.98**	0.86	0.97	85.15%
KC2	1.9	0.85	0.94	82.56%	KC2	0.09	0.86	0.94	83.90%
PC1	4.84	0.94	0.98	**93.5%**	PC1	0.31	0.94	0.98	**93.51%**

4.4.3. Logistic Classifier

The second ML classifier used in this research work was Logistic from SVM. The parameters that the researcher had applied for the classifier with correspondence value shown in Table 4.6 below under experiment 2.

Experiment 2

The key aim of this experiment was to evaluate the performance of Logistic with default parameters and all attributes applied to the four datasets. The researcher was built eight models using 10-fold cross validation test mode**.**

Table 14:- 4.6 Logistic Classifier results with all default attributes and after reduction of attributes.

Models	***Number of Instance***	***CI***	***ICI***	***Time(s)***	***Precision***	***Recall***	***In accuracy***	***Accuracy***
CM1, 22	498	439	59	0.16	0.91	0.96	11.84%	88.15%
CM1, 8	498	448	50	0.01	0.91	0.98	13.45%	89.54%
KC1,22	2109	1807	302	0.3	0.87	0.97	14.31%	85.60%
KC1, 9	2109	1797	312	0.13	0.86	0.97	14.79%	85.20%
KC2, 22	522	436	86	0.02	0.86	0.94	16.47%	83.52%

KC2,6	522	438	84	0	0.87	0.93	16.09%	83.90%
PC1, 22	1109	1025	84	0.3	0.93	0.98	7.57%	**92.42**%
PC1, 7	1109	1032	77	0.09	0.93	0.99	6.94%	**93.05**%
10-fold Cross Validation								

Table 4.6 shows the results of experiment 2. With CM1 dataset, out of 498 instances, 439 were correctly classified and the remaining 59 (11.84 %) instances were incorrectly classified and, precision and recall of the model was 0.91 and 0.96 respectively. In addition to this CM1 dataset after attributes reduction, out of 498 records 448 were correctly classified and the remaining 50 (13.45%) instances were incorrectly classified and, precision and recall of the model was 0.91 and 0.98 respectively. With KC1 dataset, out of 2109 records, 1807 were correctly classified and the remaining 302 (14.31%) instances were incorrectly classified and, precision and recall of the model was 0.87 and 0.97respectively. In addition to this KC1 dataset after attributes reduction, out of 2109 records 1797 were correctly classified and the remaining 312 (14.79%) instances were incorrectly classified and, precision and recall of the model was 0.86 and 0.97 respectively. With KC2 data set, out of 522 instances, 436 were correctly classified and the remaining 86 (16.47%) instance were incorrectly classified and, precision and recall of the model was 0.86 and 0.94 respectively. In addition to this KC2 dataset after attributes reduction, out of 522records 438 were correctly classified and the remaining 84 (16.09%) instances were incorrectly classified and, precision and recall of the model was 0.87 and 0.93 respectively. With PC1 data set, out of 1109 instances 1025 were correctly classified and the remaining 84 (7.57%) instance were incorrectly classified and, precision and recall of the model was 0.93 and 0.98 respectively. In addition to this PC1 dataset after attributes reduction, out of 1109 records 1032 were correctly classified and the remaining 77 (6.94%) instances were incorrectly classified and, precision and recall of the model was 0.93 and 0.99 respectively.

4.4.3.1. Logistic classifier Evaluation

Table 4.7 Evaluation of data sets, before preprocessing and after attribute reduction by logistic classifier. In experiment two concerning the time taken, the lowest time has seen in KC2 data set as well as the highest time taken by KC1 .As the number of attributes reduced the time taken is also decreased. In general the total time taken to accomplish the model through all four data sets before preprocessing and

after attributes selection is about 0.79 seconds and 0.23 seconds respectively. PC1 data set has almost limited to zero in this case. Additionally ,as we seen in KC1 data set as precision increase or equal in data set the accuracy also increases and when precision decrease the accuracy of data set will be decrease .In this experiment PC1 has scored the highest accuracy which is 93.05%. The parameters that the researcher applied for the classifier with correspondence value shown in Table 4.7 below under experiment 2.

Table 15:- 4.7 Evaluation of data before preprocessing and after attribute reduction by Logistic

Evaluation of data before Attributes Reduction					***Evaluation of data after Attributes Reduction***				
Datasets	***Time taken(s)***	***Precision***	***Recall***	***Accuracy***	***Datasets***	***Time taken(s)***	***precision***	***Recall***	***Accuracy***
CM1	0.16	0.91	0.96	88.15%	CM1	0.01	0.91	0.98	89.54%
KC1	0.3	0.87	0.97	85.60%	KC1	0.13	0.86	0.97	85.20%
KC2	**0.02**	0.86	0.94	83.52%	KC2	**0**	0.87	0.93	83.90%
PC1	0.31	0.93	0.98	**92.42%**	PC1	0.09	0.93	0.99	**93.05%**

4.4.3.2. Naive Bayes Classifier

The third classifier used in this research work was Naive Bayes (NB) classifier from Bayes.

Experiment 3

Table 16:- 4.8 NB Classifier result with all before preprocessing and after attribute reduction.

Models	***Number of Instance***	***CI***	***ICI***	***Time(sec)***	***precision***	***Recall***	***In accuracy***	***Accuracy***
CM1,22	498	425	73	0.02	0.92	0.91	14.65%	85.34%
CM1, 8	498	431	67	0	0.924	0.927	13.45%	86.54%
KC1,22	2109	1737	372	0.19	0.88	0.90	17.63%	82.36%
KC1,6	2109	1738	371	0.11	0.88	0.90	17.59%	82.40%
KC2,22	522	436	86	0.05	0.86	0.94	16.47%	83.52%
KC2,9	522	438	84	0.11	0.88	0.90	16.09%	82.52%
PC1, 22	1109	436	84	0.05	0.86	0.94	16.47%	83.52%

PC1,7	1109	989	120	0.02	0.94	0.93	10.82%	**89.17%**
10-fold Cross Validation								

Table 4.8 shows the results of experiment 8. With CM1 dataset, out of 498 instances, 425 were correctly classified and the remaining 73 (14.65%) instances were incorrectly classified and, precision and recall of the model was 0.92 and 0.91 respectively. In addition to this CM1 dataset after attributes reduction, out of 498 records 431 were correctly classified and the remaining 67 (13.45%) instances were incorrectly classified and, precision and recall of the model was 0.924 and 0.927 respectively. With KC1 dataset, out of 2109 records, 1737 were correctly classified and the remaining 372 (17.63%) instances were incorrectly classified and, precision and recall of the model was 0.88 and 0.90 respectively. In addition to this KC1 dataset after attributes reduction, out of 2109 records 1738 were correctly classified and the remaining 371 (17.59%) instances were incorrectly classified and, precision and recall of the model was 0.88 and 0.90 respectively. With KC2 data set, out of 522 instances, 436 were correctly classified and the remaining 86 (16.47%) instance were incorrectly classified and, precision and recall of the model was 0.86 and 0.94 respectively. In addition to this KC2 dataset after attributes reduction, out of 522 records 438 were correctly classified and the remaining 84 (16.09%) instances were incorrectly classified and, precision and recall of the model was 0.87 and 0.93 respectively. With PC1 data set, out of 1109 instances 989 were correctly classified and the remaining 120 (10.82%) instance were incorrectly classified and, precision and recall of the model was 0.94 and 0.93 respectively. In addition to this PC1 dataset after attributes reduction, out of 1109 records 989 were correctly classified and the remaining 120 (10.82%) instances were incorrectly classified and, precision and recall of the model was 0.94 and 0.93 respectively.

4.3.3.3. Naive Bayes Classifier Evaluation

In this experiment, in terms of time taken for all data sets the time decreased from 0.31 seconds to 0.24 seconds before preprocessing and after attributes selection. The lowest time taken has seen in CM1 data set as well as the highest time take has occurred in KC1 and KC2 data sets, But in PC1 data set it almost limit to zero. In general from the above four data sets the highest accuracy was occurred in PC1 which is 89.17%.

Table 17:- 4.9 Evaluation of data sets before preprocessing and attribute reduction by NB classifier.

Evaluation of data before Attributes Reduction					*Evaluation of data after Attributes Reduction*				
Datasets	***Time taken(s)***	***precision***	***Recall***	***Accuracy***	***Datasets***	***Time taken(s)***	***precision***	***Recall***	***Accuracy***
CM1	0.02	0.92	0.91	85.34%	CM1	0	0.924	0.927	86.54%
KC1	0.19	0.88	0.90	82.36%	KC1	0.11	0.88	0.90	82.40%
KC2	0.05	0.86	0.94	83.52%	KC2	0.11	0.88	0.90	82.52%
PC1	0.05	0.86	0.94	83.52%	PC1	0.02	0.94	0.93	**89.17**%

Table 18:- 4.10 Summary of all 24 experiments in one table.

Summary Evaluation by Naïve Bayes Tree							
Evaluation of data before Attributes Reduction				Evaluation of data after Attributes Reduction			
Datasets	Precision	Recall	Accuracy	Datasets	precision	Recall	Accuracy
CM1	0.98	0.93	**88.35%**	CM1	0.90	0.98	**88.755%**
KC1	0.86	0.97	**85.01%**	KC1	0.86	0.97	**85.15%**
KC2	0.85	0.94	**82.56%**	KC2	0.86	0.94	**83.90**
PC1	0.94	0.98	**93.5%**	PC1	0.94	0.98	**93.51%**
Summary Evaluation by Logistic							
Evaluation of data before Attributes Reduction				Evaluation of data after Attributes Reduction			
Data sets	Precision	Recall	Accuracy	Datasets	precision	Recall	Accuracy
CM1	0.91	0.96	**88.15%**	CM1	0.91	0.98	**89.54%**
KC1	0.87	0.97	**85.60%**	KC1	0.86	0.97	**85.20%**
KC2	0.86	0.94	**83.52**	KC2	0.87	0.93	**83.90**
PC1	0.93	0.98	**92.42%**	PC1	0.93	0.99	**93.05%**
Summary Evaluation by Naïve Bayes							
Evaluation of data before Attributes Reduction				Evaluation of data after Attributes Reduction			
Datasets	Precision	Recall	Accuracy	Datasets	precision	Recall	Accuracy
CM1	0.92	0.91	**85.34%**	CM1	0.924	0.927	**86.54%**
KC1	0.88	0.90	**82.36%**	KC1	0.88	0.90	**82.40%**
KC2	0.86	0.94	**83.52%**	KC2	0.88	0.90	**82.52%**
PC1	0.86	0.94	**83.52%**	PC1	0.94	0.93	**89.17%**

4.5. Experimental Discussion

These sections summaries the experimental results discussed above in section 4.4.2, In order to develop the software defect predictive model using single classifier which includes three classifiers experiments separately for each classifier and data sets. A total of 24 experiments were conducted based on all selected attributes. The experiment designed to examine whether number of attributes reduction in data set is improves or increase the performance of selected algorithms, to evaluate the effective algorithms by performance and to compare the performance of algorithms within each other as well as with in the earlier software defect prediction researchers.

4.5.1.Common Attributes Selection From Experiments After feature reduction

As we can see from table 4.2 above there are about 22 attributes for each and every selected data sets. But in addition to this when we see table 4.3 the reduction of attributes has expressed and the maximum attributes is 9. After discussion with different domain experts, the researcher has selected three attributes according to table 4.11 below. These are L, Uniq-Opnd and lOCBlank. Since PC1 has high accuracy the researcher was depend on it while developing the prototype.

Table 19:- 4.11 Reduced attributes from 22 to different sizes.

		Selected NASA Datasets Amount of their attributes after reduction			
R.No.	***Attributes in common***	***KC1,9***	***CM2,8***	***PC1,7***	***KC2,6***
1	V	V	Loc	V(g)	V(g)
2	D	D	Iv(g)	L	L
3	L	L	L	LOComment	B
4	LOCode	lOCode	lOComment	lOCode And Comment	Uniq-op
5	lOComment	lOComment	lOBank	LOBlank	Uniq-opnd
6	LOBlank	lOBlank	Uniq-op	uniq_Opnd	Defect
7	uniq_Opnd	uniq_Opnd	Uniq-Opnd	Defects	
8	branchCount	branchCount	Defect		
9	Defects	Defects			

4.5.2. Effect of Attribute Reduction on Dataset performance

As the researcher conducted in experiments above the performance of different algorithm were as the number of attributes decreases the performance increases. The effect of number of attribute in this study shows that the decreases number of attributes increases the precision, recall, correctly classified instances and accuracy of the model, whereas incorrectly classified instances and time taken to build model decreased due to decreasing number of attributes. In figure 4.3 shows the accuracy of predictive model for each algorithm with respect to each attributes.

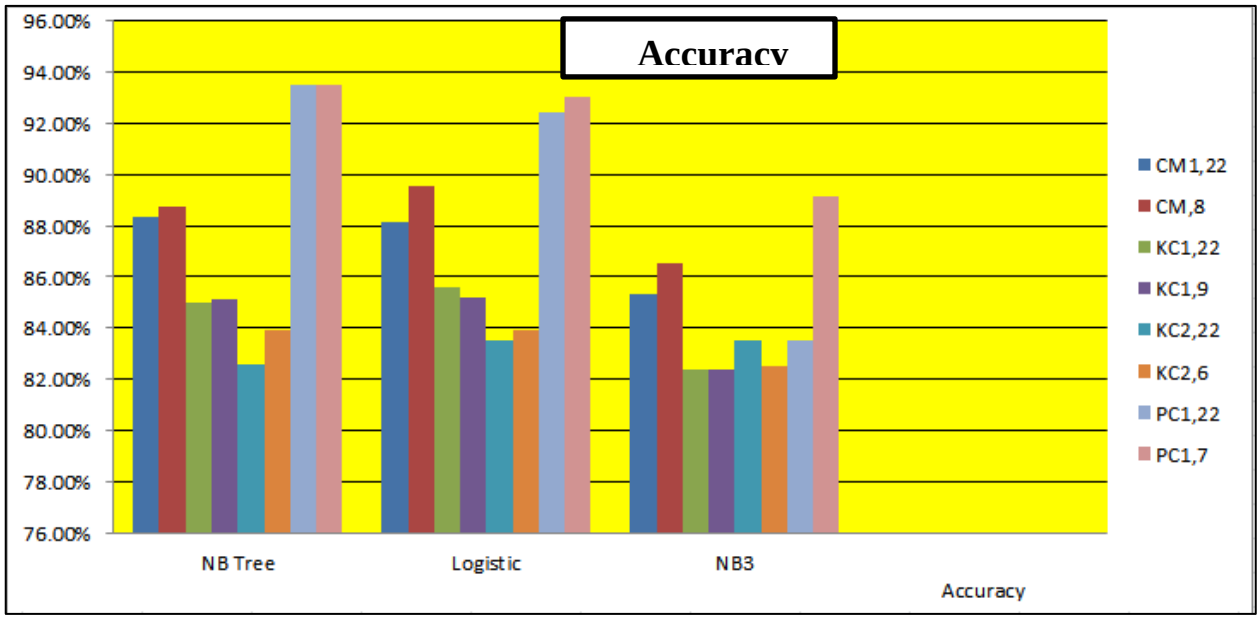

Figure 12:- 4.5 Effect of attribute reduction on classification accuracy.

4.5.3. Effect of methods on performance

In all experiments of default attributes classifiers in NB for software defect classification the experiment show that the minimum accuracy on KC2 was 82.56% on other datasets it was show more than 85%. It also showed the highest accuracy from all the remain methods. In addition to this the amount accuracy of KC2 has low in experiment of logistic classification method, when compare to the give datasets. It was scored 83.52% and also again less than 85% of accuracy, whereas, the remains datasets had more than 85% including PC1 which is has highest accuracy of 89.17%. As a general when the experimenter compare all the three methods used in Naive Bayes, only one dataset has accuracy more than 85% that was CM1 which scored about 85.34% .The remain datasets had

below 85% of accuracy. Even the minimum accuracy of all from the three methods was found in this technique about 82.36% which was occurred by KC1 dataset.

4.5.4. Model comparison from point view of experiments result

The researcher has to identify which machine learning algorithm performs best in predicting software defect. In the earlier section the researcher showed the experiments to select machine learning algorithms and to build the model by better algorithm from above, however the researcher did not select which model is best for software defect prediction. As we have seen basically, about 24 experiments were gone under selected ML classification methods by considering two things, before preprocessing and after attributes reduction datasets. In this section the researcher has selected the best model based on the performance measures that were putted in section 3.7 of this paper as methodology part. In order to compares those methods the researcher used various performance measures like accuracy, Precision and Recall. Model comparison in experiment was considered PC1 for each experiment in this case, Because of the maximum accuracy results were scored within this dataset as shown in the table 4.10 above.

Additionally, the researcher has made another performance measurement in order to check which algorithms produce better or even best result. To come up with this comparison the researcher has compare only two models NB Tree classifier from decision tree and Logistic classifier from SVM by ignoring NB classifier from Bayes which is the third parts of the study. Because the NB classifier shows the lowest accuracy when compare to the two selected models. Taking this reason into mind the researcher goes ahead comparing the developed model NB Tree classifier and Logistic classifier by their sensitivity (true positive rate), specificity (true negative rate), time taken to accomplish and accuracy again additionally confusion matrix. The highest accuracy model in this paper was generated in NB Tree classifier in experiment 1 which scored 93.5%.The lowest accuracy of models in PC1 dataset this study was generated in experiment 3 which was about 89.17%.The researcher also consider the execution time to develop the model.

Table 20:- 4.12 Performance of selected models (NBTree and Logistic) on PC1 datasets attributes.

Models	*Time take (seconds)*	*Specificity*	*Sensitivity*	*Accuracy*
NBTree before preprocessing attributes of PC1 ,22 data set	4.84	0.97	0.291	**93.5%**
NBTree after attributes reduction of PC1,7 dataset	0.31	0.98	0.236	**93.51%**
Logistic before preprocessing attributes of PC1 ,22 data set	0.3	0.99	0.178	**92.42%**
Logistic after attributes reduction of PC1,7 data set	0.09	0.99	0.103	**93.05%**
NBTree before preprocessing attributes of PC1,22 data set	0.09	0.94	0.94	**83.52%**
NBTree after feature selection of PC1,7 data set	0.02	0.98	0.98	**89.17%**

As accuracy is used to measure the proportion of the correctly classified modules to the total modules. The researcher has tried to express the PC1 accuracy from the above experiments.

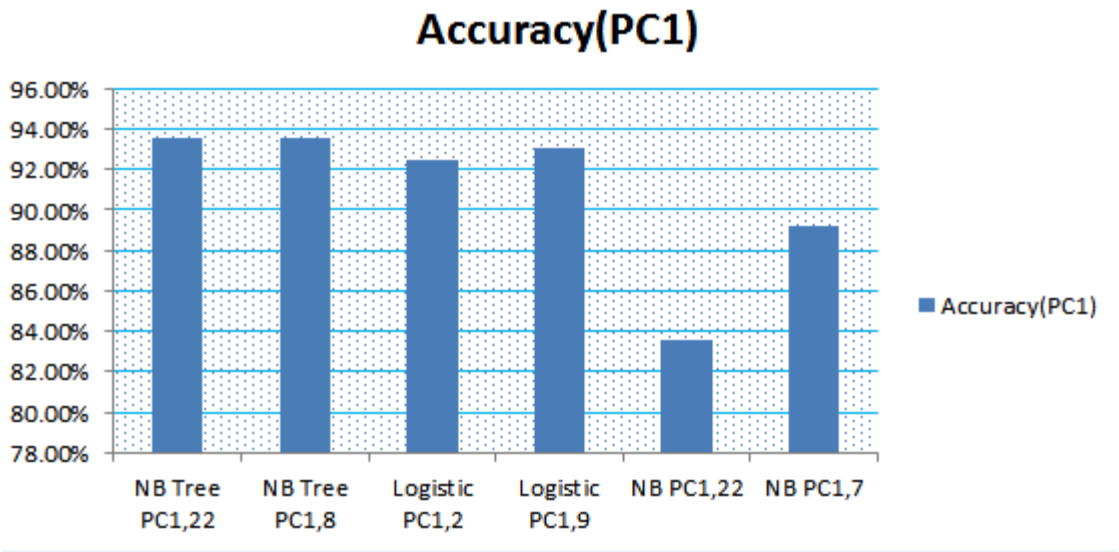

Figure 13:-4.6 Models comparison with PC1data set.

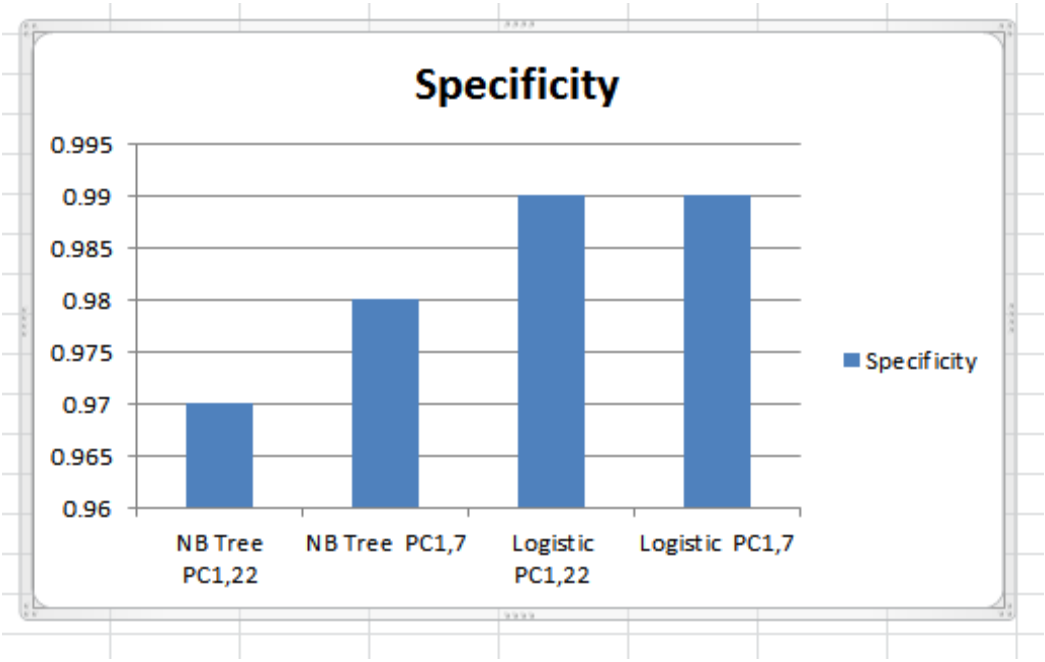

Figure 14:- 4.7 Specificity diagram of PC1 data set.

Specificity describe the “TRUE POSITIVE” which is directly connected with the recall. According to this study, PC1 data set has less error prone and more recall that means specificity in this data set become high.

Table 21:- 4.13 Confusion matrix's of NB Tree in PC1 Data set

		prediction class		Total
		False	True	
Actual class	False	1020(TN)	12(FP)	1032
	True	60 (FN)	17(FP)	77
Total		1080	19	1109

As shown in table 4.12, in NB classifier the correctly classified instance of PC1 dataset was 1020+17=1037 (93.5%) and incorrectly classified instance 60+12=72 (6.5%). The specificity of the model was 1020/1037=0.98.

Table 22:- 4.14 Confusion matrix by Logistic in PC1 dataset

		prediction class		Total
		False	True	
Actual class	False	1020(TN)	12(FP)	1032
	True	72(FN)	5(FP)	77
Total		1092	17	1109

As shown in table 4.13 in Logistic classifier the correctly classified instance of PC1 dataset was 1020+5=1025 (92.42%) and incorrectly classified instance 72+12=84 (6.5%). The specificity of the model was 1020/1025=0.99.

4.6. Rule Extraction from Naive Bayes Tree

In this Study, based on the performance evaluator like Accuracy tool; Naive Bayes Tree Classifiers achieved relatively highest performance when we compared with other two classifiers, that we used

in this study. As we have seen in appendix 3 Naive Bayes Tree generated 6 rules for predicting software fault. The researcher selected 4 rules that cover most of instances within the given data set, and then the researcher made deep discussion with domain experts in order to assure that the selected rules really works for all instances. Selected rule listed and discussed below.

Rule 1:- IF L < =30.915 and Unique_Opnd <=5.5 and L<=17.235: **Then** software product = False. Software codes with line of code less than and equal to 30.915, Unique_Opnd less than and equal to 5.5 and line of code less than and equal to 17.235: **Then** software product to false.

Rule 2:- IF L < =30.915 and L>17.235: **Then** software product = False. Software codes with line of code less than and equal to 30.915 and line of code less than and equal to 17.235: **Then** software product to false.

Rule 3:- IF L > 30.915 and Unique_Opnd >=72.5 and IO_Blank > 6.5: **Then** software product = True. Software codes with line of code greater than and equal to 30.915 Unique_Opnd greater than 72.5 and IO_Blank greater than 6.5: **Then** software product to True.

Rule 4:- IF L > 30.915 and IO_Blank > 6.5: **Then** software product = True. Software codes with line of code greater than and equal to 30.915 and IO_Blank greater than 6.5: **Then** software product to True.

4.7. Model comparison with related works

As far as the knowledge of the researcher there is no such maximum accuracy up now as reviewed so many literatures documents. PC1 data set was scored the maximum amount of accuracy in this study during experimentation processes ,as a result the researcher has compares with the earlier studied by different researchers**.**

Table 23:- 4.15 Model comparison with related works

Role No	***Author name (year)***	***Objective of the study***	***Datasets used By Researches***	***Methodology/approaches/ Techniques/tools/ used***	***Key findings***
1	V.Vashisht et. al (2015)	-To improve effectiveness and acceptance of framework	Real data from different software companies	Neural Network (NN)	-Accuracy 90%

2	Shaik Nafeez Umar, (2013)	To wise project better defect and prediction	Used 20 past release datasets.	-Multiple linear regression models.	Accuracy 84%.
3	Ermias Berhanu (2016)	Developing model for Software complexity problem testing	CM1, JM1, KC2,KC1 and KC3.	-10 fold cross validation	-Accuracy for KC3 was about 90.82%
4	Current study (December, 2017)	developing model for software defect prediction: by feature reduction	PC1, KC1, CM1 and KC2	-10 fold cross validation	-Accuracy of PC1 data set **93.51%**

4.8. Software defect Classification System: Framework

The researcher was develop a frame work which is known as software defect classification system that is used to classify a software product into one of the software labels (defect or non-defect) according to rules that has mentioned in section 4.6 above.

Software defect classification system includes the user to load extracted source code data from data base when the user clicks on load data button, predict the user based on the trained in to defect or not defected one, if we click on predict button. From figure 4.6 there are 22 inputs (attributes) which selected from four NASA data set and output is calculated by click on the predict button. Predict button is used to categorizing the input data in to fault or non-fault one. Exit button is used to close out from the graphical user interface.

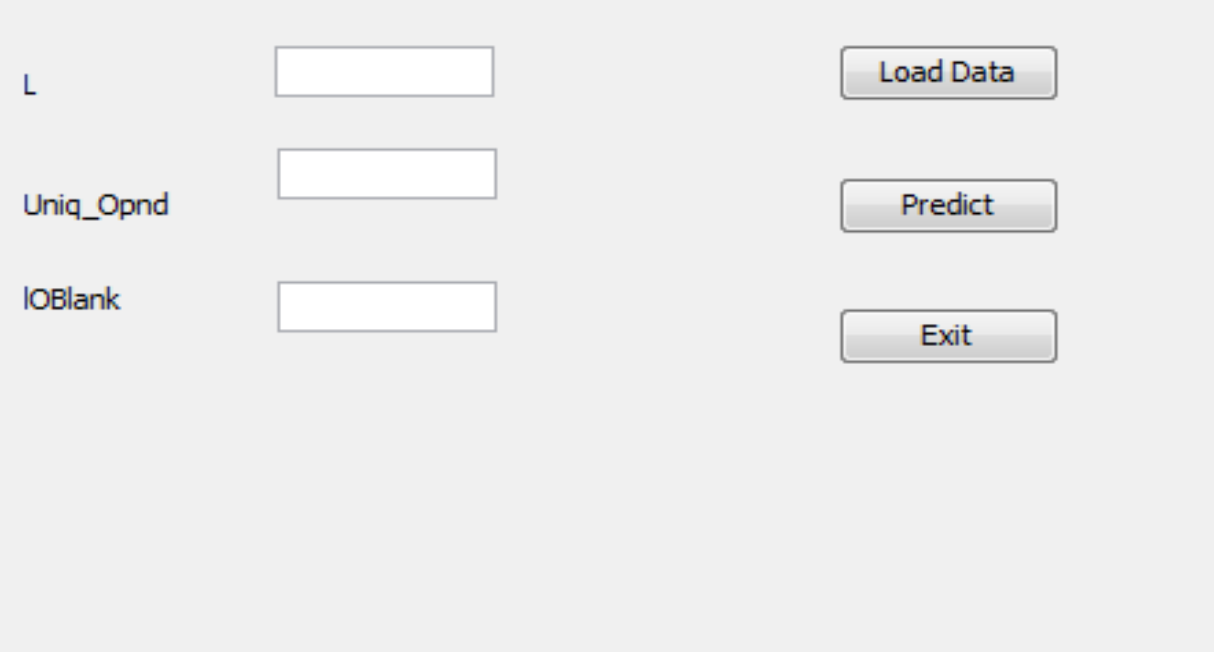

Figure 15:- 4.8 Software defect classification system user interface.

As shown Figure 4.7 below, this prototype prediction model can be used for predicting software faults based on the rules generated by NBTree classifier. The researcher has loaded the extracted source code data in number of line of code, to the prototype as we can see in Figure 4.7 to show the predicted result of software faults as TRUE based on rule in section 4.6. As shown in appendix 3, the researcher has used C programming language to predict software defect according to the selected rules that generated from NB Tree and suggested by different domain experts.

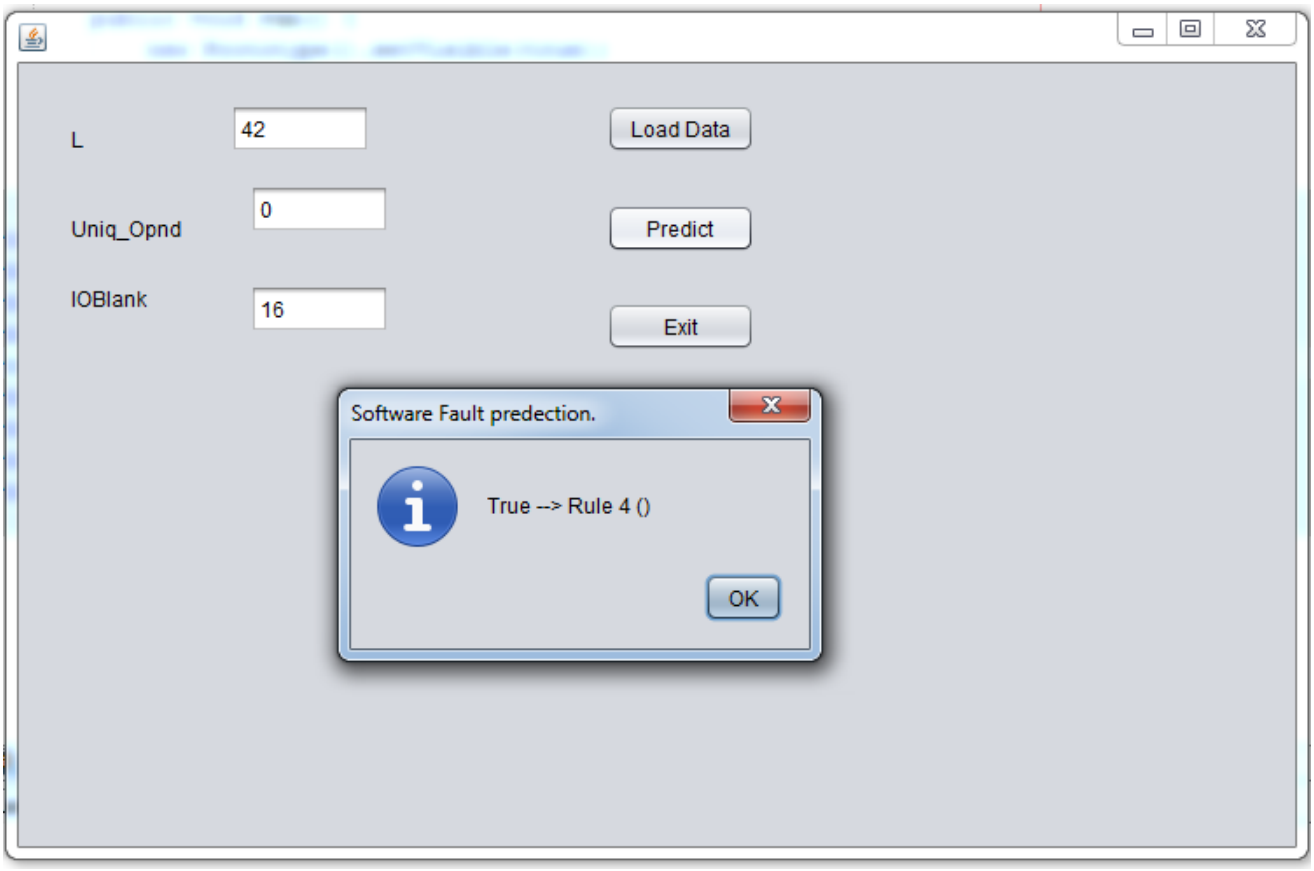

Figure 16:- 4.9 Software defect prediction prototype user interface with sample result.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

5.1. Conclusion

Software's are used almost everywhere and in every tread of our life. Many software companies are building different huge systems within large software data's for various purposes; to coverage the necessity of software within short time and better quality to their customers Software companies have various measurement methods in order to check their software quality. To provide or in order to serve the quality software for users, by any means software testing is must. Testing a large or huge data set can take time and cost, because huge data sets contain so many features . To come up with this solution, applying machine learning is desirable by developing a model to make easy a huge software data for developers in order to test. In machine learning, feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

In this research, the methodology has used where different machine learning classification techniques approach such as, NBTree, Logistic and NB classifiers were used with four selected NASA MDP data sets. The four data sets CM1,KC1,KC2 ,PC1 and their modules 498, 2109,522 ,1109 respectively were used for operating experiments .All the selected data sets have 22 attributes each. To prepare the data set for experiments have been takes place for building of the model by using NB Tree, Logistic and NB classification in machine learning algorithms.

The researcher has done 22 experiments using four data sets and three classification algorithms by weka machine learning software tool and come up with building different models. After feature (attributes) selection the four selected data set reduced from 22 each to 8, 9, 6, 7 and CM1, KC1, KC2 and PC1 respectively.

For the performance evaluation of built models 10-fold cross validation such as, Accuracy, Precision, recall and time to perform were to evaluate the performance of model. In this research experiment the researcher has get with the consideration of the size of the software data set decreases there is an effect of method of performance predictive model. As a result the experiment shows that when the size of datasets attributes decreases the performance of the machine learning algorithms increases, this was due to decreases or reduces of the size of software datasets scalability of the model.

In this study the effect of method to the performance of the machine learning algorithms high accuracy was achieved by the NB Tree in PC1 data set before preprocessing as well as and after feature reduction and based on performance evaluator the best algorithm was NB Tree classifier. Which has scored Accuracy **93.51%.**In addition to this the researcher has selected similarities of the reduced attributes from experiments, which has maximum 9 attributes. Depending on the NBTee algorithm model attributes about six rules has generated and only four rules has selected to develop the graphic user interface using neat beans for software fault prediction system.

5.2. Future work

This study was dedicated on testing and comparing machine learning techniques performance to build best defect proneness models using NASA MDP datasets. The researcher recommended the following future works to be continued for any new researcher who have motivated

1. Using Rapid Miner tool rather than Weka for machine learning techniques performance to build best software defect prediction model.

2. Predict software defect Using Eclipse data code rather than NASA datasets.

References

1. A.Harry. (2015). "Machine learning capabilities, limitation and implications: echnology Futures."

2. Akiyama. (1971). "An Example of Software System Debugging." In Proceedings of the software testing confrence.

3. Arthur H and Thomas J. (September,1996)." Structured Testing:A Testing Methodology using the Cyclomatic." National Institute of Standards and Technology.

4. Aspree K and Parvinder S. (2011)." A k-means Based Approach for Prediction of Level of Severity ". Proceedings of International conference on Intelligent.

5. Ayúe Tosun, Ayúe Bener. (2009)." Reducing False Alarms." The third International Symposiumm on.

6. Bermingham and Pong-Wong et.al. (2015)." Application of high-dimensional feature selection: evaluation for genomic prediction in man."

7. Bharat Deshmukh,Ajay S.Patil & B.V.Pawar. (2011, December)."Comparison of Classification Algorithms using." Vol. 4, pp. 85-90.

8. Bhatti, H. R.. "Automatic Measurement of Source Code Complexity." master's thesis Computer Science and Engineering Luleå University of Technology.

9. Bieman,J.M. (1997). "Software Metrics: A Rigorous & Practical Approach":IBM Systems.

10. Boehm,B.,& Basili,V.R. (2001). "Top 10 list software." 34(1), pp. 135–137.

11. Catal, Cagatay. (2012)." Performance Evaluation Metrics for Software.": Istanbul Kultur University, Department of Computer Engineering, Atakoy.

12. Cem Kane,J. D.,Hendrickson, E.,& Smith-Brock,J. (2001). "Managing the proportion of testers to (other) developers".

13. Christian M,Gail K,and Marta A. "An Approach to Software Testing of Machine Learning."

14. Clancy, T. (2014). The Standish Group Report CHAOS " Project Smart.".

15. D.Gray,D.Bowes,N.Davey,Y.Sun,and B.Christianson. (2012). "Reflections on the NASA MDP data." vol.6, pp. 549-558.

16. D.Tomar and S.Agarwal. (2013). "A survey on Data Mining approaches for Healthcare". International Journal of Bio-Science and Bio-Technology, 5, pp. 241-266.

17. D'Ambros,M.,Lanza,M.,& Robbes,R. (2012)." Evaluating defect prediction. Empirical Software".

18. Damtew, A. (2011). "Designing a predictive model for heart disease detection using datamining techniques". Msc thesis school of public health.Addis Ababa.

19. Durgesh k and lekha b. (2009). "Data Classification Using Support Vector Machine". Journal ofTheoretical and Applied Information Technology, pp.1-7.

20. Ekbal R et.al. (2012, September)."A Survey in the Area of Machine Learning and Its Application".*37*, pp. 1-7.

21. Ermiyas, B. (2016)."software complexity predection model: a combined machine learning approch". PP. 82-83.

22. Gayathri M, A. Sudha. (2014, May)."Software Defect Prediction System using". International Journal of Recent Technology and Engineering (IJRTE),3(2), pp.1-6.

23. Gehrke,J.,Ramakrishnan,R.,Ganti,V. (1998)."RainForest - a Framework for Fast Decision Tree Construction of Large Datasets". New York,USA.

24. George T,Ioannis K,Ioannis P,and Ioannis V. (2006)."Modern Applications of Machine Learning". Proceedings of the 1st Annual SEERC Doctoral Student Conference, vol.1-10.

25. Geralds, J. (2002). "Software-Test management". in USA.

26. Guyon, I. (2003)."An Introduction to Variable and Feature Selection". Journal of Machine Learning Research.

27. H.B.KLASK. (2003). "A Study of Software Metrics". Graduate Program in Electrical and Computer.

28. H.Shivkumar. (2012, October). "Software Testing Techniques". 2, pp. 433-439.

29. Hall,M and Hall,M.A. (1999). "Correlation-based feature selection for machine learning". Doctoral dissertation.

30. Halstead, M. (1997). "Elements of Software Science (Operating and Programming". New York, NY, USA: Elsevier Science

31. Hamon and Dhaenens. et.al.(2013, November)."Feature selection in high dimensional regression problems for genomics".

32. IEEE. (1990)." Glossary of Software Engineering". IEEE Standard 610.12-1990.

33. J, C. (March 2006)."Strengths and Weaknesses of Software Metrics". Software Productivity Research LLC.

34. Japkowicz,N.,& Shah,M. (2011). "Evaluating learning algorithms: a classification perspective". Cambridge University Press.

35. Jianing Shi, Wotao Yin et.al. (2010)." Fast Hybrid Algorithm for LargeScale 1-Regularized Logistic Regression". Journal of Machine Learning Research, pp. 713-741.

36. Jiliang Tang et.al. (2014). Feature Selection for Classification: A review.

37. Jing and et...al.(2014)."Dictionary learning".(Paper presented at the Proceedings of the 36th).

38. Jobaer Islam Khan,Alim Ul Gias,Saeed Siddik and Saeed Siddik. (2014, May)."An Attribute Selection Process for Software".

39. Jones,C.,& Bonsignour,O. (2012)."The Economics of Software Engineering". pearson Education.1, p. 33.

40. Jong and Marchioret.al. (2004). "Feature selection in proteomic pattern data with support vector machines". In Computational Intelligence in Bioinformatics and Computational Biology, 41- 48.

41. Kennedy, Kenneth. (2013)."Credit Scoring Using Machine Learning". Doctor of Philosophy, Dublin.

42. Kim et.al. (2011)."Dealing with noise in defect prediction". Paper presented at the Software Engineering (ICSE)33rd International Conference on.

43. Lieberman, H. (1997)."The debugging scandal and what to do about it". 40(4), pp. 26 –29.

44. M.Dash and H.Liu. (1997)."Feature selection for classification". Intelligent data analysis".

45. M.Ruchika. (November2015)."A systematic review of machine learning techniques for software fault prediction". Applied Soft Computingpp. 504 –518.

46. M.Saini and K.Kaur. (2014)."A Review of Open Source Software Development Life Cycle Models". International Journal of Software Engineering and Its Applications.

47. Malkit S and Dalwinder S. (2013, May)." Software Defect Prediction Tool based on Neural Network". International Journal of Computer Applications, 70 , pp. 22-28.

48. McCabe, T. (1976)."A complexity measure". Software Engineering. IEEE Transactions.

49. Menzies,et.al. (2004)."Assessing predictors of software defects". Paper presented at the Proc. Workshop Predictive Software Models.

50. Mikyeong P and Euyseok.H. (2014)."Software Fault Prediction Model using Clustering Algorithms Determining the Number of Clusters Automatically". International Journal of Software Engineering and Its Applications, 8, pp. 199-204.

51. Misha Kakkar and Sarika Jain. (2015)." Feature Selection in Software Defect: A Study Comparative".

52. N.Gayatri,S.Nickolas,A.V.Reddy. (2010, October ,). "Feature Selection Using Decision Tree Induction". Proceedings of the World Congress on Engineering and Computer Science, 1, pp. 20-22.

53. Naheed Azeem,Shazia Usman. (2011, November). "Defect Prediction Leads to High Quality Product". Journal of Software Engineering and Applications, 4, pp. 639-645.

54. Nam,J.,Pan,S.J.,& Kim,S. (2013)."Transfer defect learning". Paper presented at the Proceedings of the International Conference on Software Engineering.

55. "NASA.http://promise.site.uottawa.ca/SERepository".

56. O.L.Mangasarian and W.N.Street. (1994, January). "Breast Cancer Diagnosis and Prognosis via Linear Programming".

57. O.T.Pusatli,S.Misra. (2011). "Software Measurement Activities in Small and . Hungarica: Acta Polytechnica.

58. Pooja P and D.A.Phalke. (June 2015). "Software Defect Prediction for Quality Improvement Using Hybrid", vol.4.

59. Quinlan, J. R. (1986). "Induction of decision trees. Machine learning". 1.

60. R.Chidamber and F.Kemerer. (1994, June)." A Metrics Suite for Object Oriented Design." IEEE Transactions on, 20, pp. 476-493.

61. Rajender., S. a. (2012, August). "Case Studies of Most Common and Severity Types of Software". *2*, pp. 341-347.

62. Rajkumar G and K.Alagarsamy. (2013, January). "The Most Common Factors For The Failure Of Software". 11, 74-77.

63. Rasneet K and Iqbal S.(2014, August)."Latest Research and Development on Software Testing Techniques and Tools". International Journal of Current Engineering and Technology, 4.

64. RevaITM, P. G.."Basics of Software Testing".Bangalore: VTU E-Learning.

65. Richard, S. (2014)."SDLC and Development Methodologies". Global Journal of Computer Science and technology.

66. Romi Satria Wahono and N.Suryana Herman. (January,2014). "Genetic Feature Selection for Software Defect". American Scientific Publishers.

67. S.Aruna,S.P.Rajagopalan,and L.V.Nandakishore. (2011, August)."An Empirical Comparison of Supervised". International Journal of Information Technology, 1, pp. 81-92 .

68. S.D.Conte et.al. (1987). A Software Metrics Survey. Purdue University.

69. S.Misra. (2011). Evaluation Criteria for Object-oriented Metrics. Hungarica: ActaPolytechnica.

70. Sahana, D. C. (2013). Software Defect Prediction Based on Classication Rule Mining.

71. Saiqa.A, L. F. (2015, May)."Benchmarking Machine Learning Techniques for Software Defection". pp. 11-23.

72. Sammut,C.,& Webb,G.I. (2011). "Encyclopedia of Machine Learning." Springer.

73. Shichaozhang,C.Z.aq. (2003, May). "Data Preparation for Data Mining.",Applied Artificial Intelligence, 17(375–381).

74. Singhal,M.J.Swasti. (2013, May). "A Study on WEKA Tool for Data Preprocessing, Classification and Clustering.", pp. 250-253.

75. Smola,A.,& Vishwanathan,S.V.N. (2008). "Introduction to machine learning." pp. 32-34.

76. T.Menzies et al. (2007). "Learn defect predictors and data Mining Static Code Attributes to learn defect predictors Software Engineering", IEEE Transactions on, 2-13.

77. Taghi M. Khoshgoftaar, Kehan Gao, Naeem. (2010). Attribute Selection and Imbalanced Data:. International Conference on Tools with Artificial.

78. Tang and Zheng al...et. (2005, June). "A comparative study of medical data classification methods based on decision tree and system reconstruction analysis". 4(1), pp. 102-108.

79. Tina R and Sherekar S. (2013, April)."Performance Analysis of Naive Bayes and J48 Classification Algorithm for. International Journal Of Computer Science And Applications, *6*, pp. 256-261.

80. Tom M. Mitchell. (1997)."Machine Learning".

81. Umar, S. N. (2013, May)."software testing defect prediction model- a practical". IJRE*:* International Journal of Research in Engineering and Technology, 2(5).

82. US NIST. (2002). USA.

83. Vapnik. (1995)."Concept on support vector network".Khuwer Academic publishers pp.273-297, Boston.

84. Vashisht,V.,Lal,M.and Sureshchandar,G.S. (2015, Augest). "A Framework for Software Defect". Journal of Software Engineering and Applications, 8, pp. 384-394.

85. Vikas S and Jatinderkumar R. (2014, May). "Cataloguing Most Severe Causes that lead Software Projects to Fail ". International Journal on Recent and Innovation Trends in Computing and Communication, 2, 1143– 1147.

86. Vikas S and Jatinderkumar R. (May,2014). "Cataloguing Most Severe Causes that lead Software Projects to. vol. 2,.

87. Wahono, R. S. (2015, April)."A Systematic Literature Review of Software Defe Prediction:". Journal of Software Engineering, 1(2356-3974), pp.1-13.

88. Whittaker,J. (2000)."What is software testing? And why is it so hard?". Software. IEEE.

89. Witten,I.H.,& Frank,E. (2005)."Data Mining: Practical machine learning tools and. Morgan Kaufmann.

90. Xia C,Michael R,Kam-Fai W,and Mabel W. Compare:"A Generic Quality Assessment". Center of Innovation and Technology.

91. Zhang, D."Applying Machine Learning Algorithms In Software Development". California State.

92. Zheng,J.(2010)."Cost-sensitive boosting neural networks for software defect prediction".Expert Systems with Applications.

93. Zhou Xu et.al. (2015)."The Impact of Feature Selection on Defect Prediction Performance". china: Wuhan University.

APPENDIXES

Appendix 1: PC1 Sample dataset by ARFF file format

@relation PC1

@attribute loc numeric

@attribute v(g) numeric

@attribute ev(g) numeric

@attribute iv(G) numeric

@attribute N numeric

@attribute V numeric

@attribute L numeric

@attribute D numeric

@attribute I numeric

@attribute E numeric

@attribute B numeric

@attribute T numeric

@attribute lOCode numeric

@attribute lOComment numeric

@attribute locCodeAndComment numeric

@attribute lOBlank numeric

@attribute uniq_Op numeric

@attribute uniq_Opnd numeric

@attribute total_Op numeric

@attribute total_Opnd numeric

@attribute branchCount numeric

@attribute defects {false,true}

@data

1.1,1.4,1.4,1.4,1.3,1.3,1.3,1.3,1.3,1.3,1.3,1.3,2,2,2,2,1.2,1.2,1.2,1.2,1.4,false

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,true

91,9,3,2,318,2089.21,0.04,27.68,75.47,57833.24,0.7,3212.96,80,44,11,31,29,66,192,126,17,true

109,21,5,18,381,2547.56,0.04,28.37,89.79,72282.68,0.85,4015.7,97,41,12,24,28,75,229,152,38,true

505,106,41,82,2339,20696.93,0.01,75.93,272.58,1571506.88,6.9,87305.94,457,71,48,49,64,397,1397,942,178,true

107,25,7,14,619,4282.78,0.02,52.91,80.95,226588.75,1.43,12588.26,103,32,4,39,35,86,359,260,40,true

74,11,1,8,294,1917.93,0.03,28.77,66.66,55178.46,0.64,3065.47,60,71,14,49,29,63,169,125,21,true

602,136,123,123,2785,25942.69,0.01,105.26,246.47,2730637.23,8.65,151702.06,600,40,2,225,99,538,1641,1144,236,true

Appendix 2: C programming Code to Extract Attributes Using Prest

```
/*
 * C Program to find the first capital letter in a string using
 * Recursion
 */
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char caps_check(char *);
int main()
{
    char string[20], letter;
    printf("Enter a string to find its first capital letter: ");
    scanf("%s", string);
```

```
letter = caps_check(string);
if (letter == 0)
{
   printf("No capital letter is present in %s.\n", string);
}
else
{
   printf("The first capital letter in %s is %c.\n", string, letter);    }
   return 0;
}
char caps_check(char *string)
{
   static int i = 0;
   if (i < strlen(string))
   {
      if (isupper(string[i]))
      {
         return string[i];
      }
      else
      {
         i = i + 1;
         return caps_check(string);
      }
   }
   else return 0; }
```

Appendix 3: Extracted Rules by NB Tree

I <= 30.915

| uniq_Opnd <= 5.5

| | I <= 17.235: NB 3

| | I > 17.235: NB 4

| uniq_Opnd > 5.5: NB 5

I > 30.915

| uniq_Opnd <= 72.5

| | lOBlank <= 6.5: NB 8

| | lOBlank > 6.5: NB 9

| uniq_Opnd > 72.5: NB 10

Appendix 5: Java Code to load Data from MySQL

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         try
        {
          String dbURL = "jdbc:mysql://localhost:3306/proto1";
        String dbUser = "root ";
        String dbPass = " ";
         Class.forName("com.mysql.jdbc.Driver");
         Connection conn = DriverManager.getConnection(dbURL,dbUser,dbPass);
        //System.out.println("Hello");
         PreparedStatement pst = conn.prepareStatement("select * from data");
        ResultSet rs = pst.executeQuery();
```

```
        while(rs.next())
       {
            String a=rs.getString("L");
           String b=rs.getString("Uniq_Opnd");
           String c=rs.getString("IOBlank");
           jTextField1.setText(a);
           jTextField2.setText(b);
           jTextField3.setText(c);
       }
             }
  catch(Exception e){
     // out.println("wellcome");
      //out.println(" wrong !! Please try again");
  }
   }
```

Appendix 6: Java Code to Predict Software Product Button

```
Private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
       if (jTextField1.getText().equals(""))
       {
              JOptionPane.showMessageDialog(null, "Please Input the first ");
          } else if (jTextField2.getText().equals("")) {
           JOptionPane.showMessageDialog(null, "Please Input the second instance ");
       } else if (jTextField3.getText().equals("")) {
           JOptionPane.showMessageDialog(null, "Please Input the third instance  ");
       }
       float one = Float.parseFloat(jTextField1.getText());
       float two = Float.parseFloat(jTextField2.getText());
```

```
        float three = Float.parseFloat(jTextField3.getText());

            if (one <= 30.915 && two <=5.5 && one > 17.35) {

//            //System.out.println("False");

         javax.swing.JOptionPane.showMessageDialog(this, "False --> Rule 1 ()" , "Software Fault predection.", JOptionPane.INFORMATION_MESSAGE);

        } //Rule 2:- IF LOC_Blank <=3 and Halsted _ length >38 and LOC _ Blank <=2 and LOC_ Comment <=0 and Halsted _ Error _EST > 0.06 and Design _ Complexity <= 2: Then software product to False

        else if (one <= 30.915 && one > 17.35) {

           //System.out.println("False");

           javax.swing.JOptionPane.showMessageDialog(this, "False --> Rule 2 ()", "Software Fault predection.", JOptionPane.INFORMATION_MESSAGE);

        }

            else if (one > 30.915 && two  > 72.25 && three > 6.5) {

           //System.out.println("False");

           javax.swing.JOptionPane.showMessageDialog(this, "True --> Rule 3 ()", "Software Fault predection.", JOptionPane.INFORMATION_MESSAGE);

        }

            else if (one > 30.915 && three > 6.5) {

           //System.out.println("False");

           javax.swing.JOptionPane.showMessageDialog(this, "True --> Rule 4 ()", "Software Fault predection.", JOptionPane.INFORMATION_MESSAGE);}

            else {

              javax.swing.JOptionPane.showMessageDialog(this, "unkown case  --> Rule 5 ()", "Software Fault predection.", JOptionPane.INFORMATION_MESSAGE);

            }

// TODO add your handling code here:

 }
```