Software Fault Prediction using Machine Learning Techniques

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

Hemesh Bhardwaj

(Roll No. 801632011)

Under the supervision

of:

Dr. Ashutosh Mishra

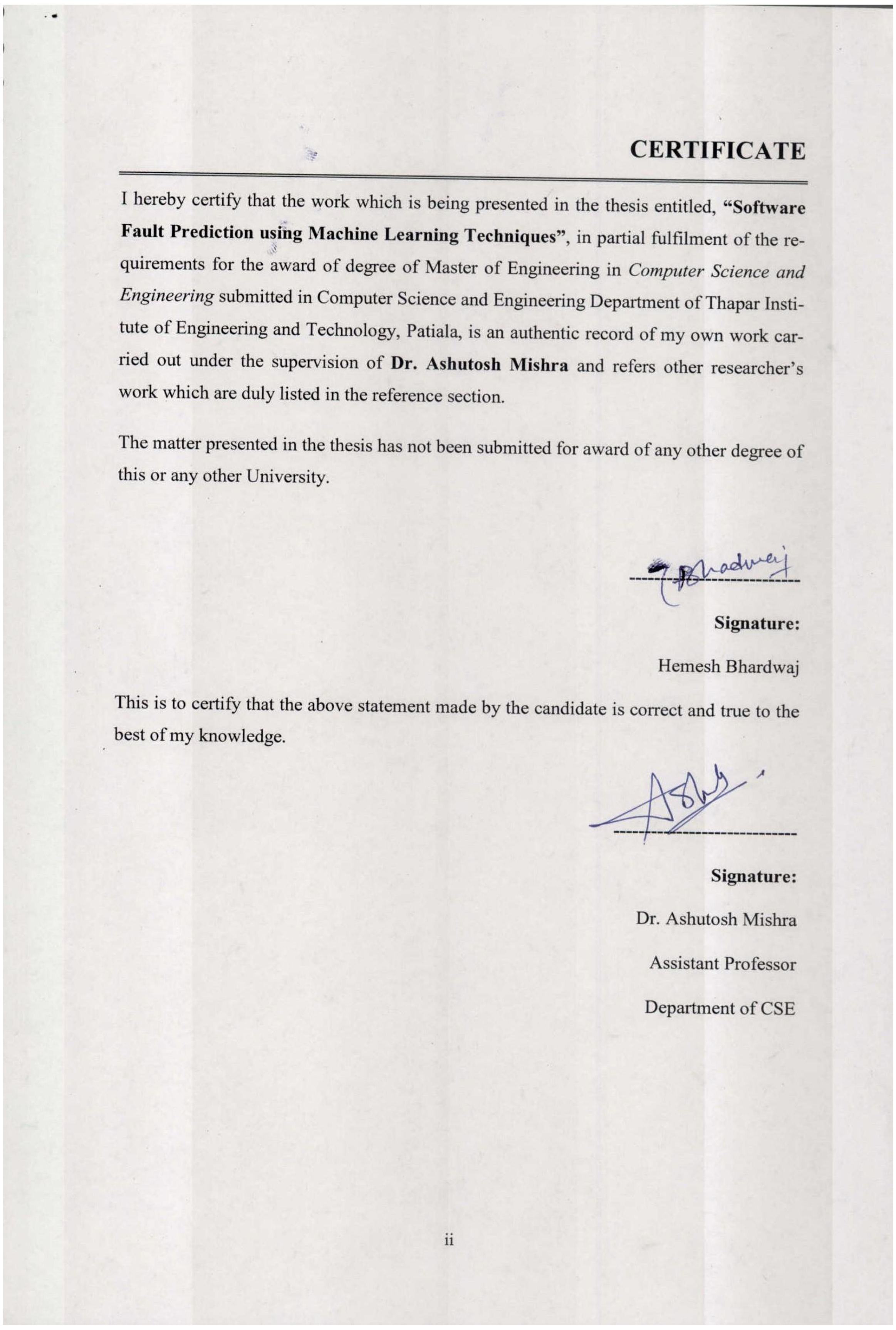
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

PATIALA – 147004

June 2018



CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Software Fault Prediction using Machine Learning Techniques"**, in partial fulfilment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Ashutosh Mishra** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

[illegible]

Signature:

Hemesh Bhardwaj

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

[illegible]

Signature:

Dr. Ashutosh Mishra

Assistant Professor

Department of CSE

ACKNOWLEDGEMENT

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.

This work would not have been possible without the encouragement and able guidance of my supervisor **Dr. Ashutosh Mishra**. I thank my supervisor for their time, patience, discussions and valuable comments. Their enthusiasm and optimism made this experience both rewarding and enjoyable.

I am equally grateful to **Dr. Maninder Singh**, Professor and Head, Computer Science & Engineering Department, a nice person, an excellent teacher and a well credited researcher, who always encouraged me to keep going with work and always advised me with his invaluable suggestions.

I am again thankful to **Dr. Ashutosh Mishra**, P.G. Coordinator, and Assistant Professor of Computer Science & Engineering Department of TIET for the motivation and inspiration for the completion of thesis.

I will be failing in my duty if I don’t express my gratitude to **Dr. S.S. Bhatia,** Professor and Dean of Academic Affairs, TIET, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at TIET memorable.

Last but not least, I would like to thank my family whom I dearly miss and without whose blessings none of this would have been possible. To my parents, I own thanks for their wonderful love and encouragement. I would also like to thank my close friends for their constant support.

Date: July, 2018 **(Hemesh Bhardwaj)**

Place: Thapar Institute of Engineering and Technology, Patiala **(801632011)**

ABSTRACT

Fault-prediction techniques are aiming towards prediction of the software modules that are faulty so that it could be beneficial in the upcoming phases of software development. Difference performance criteria are being employed in order to boost the performance of the already existing ways. However, the main issue is the perspective of compiling their performances which is ignored constantly. Classification is the most used technique that is being used for the exclusion of faulty from non-faulty modules. Previous work under this topic has been carried out using different techniques. Here, we have used tried to enhance the process by using feature selection method so as to make the prediction more accurate and less time consuming. Moreover, nine fault prediction techniques have been such as Adaboost, multilayer perceptron, decision tree regression, linear regression, Boosting, random forest, support vector machine, bagging, boosting for the prediction of number of faults. The study has been carried out using ten software project datasets from PROMISE repository. Mainly, AAE, ARE and feature selection techniques have been used to evaluate the results of the investigation. The kruskal-wallis test has been applied on the ARE and AAE values to check whether the difference of performance of the various machine learning techniques is statistically significant or not.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Several prediction approaches are being employed in Software engineering discipline such as correction cost prediction test effort prediction, quality prediction, security prediction, reusability prediction, fault prediction and effort prediction. However, the aforementioned approaches have only reached initial phase and further research is required to be steered to reach at the level of robustness. Software fault prediction is by far the most prevalent research field in these approaches for prediction and many research centres have initiated new projects in this field. Software metrics and fault data are used to predict the models that are faulty. Binary form is used to report the error, if a module is faulty than it is marked as 1 else 0. Software faults are programming errors, which cause the system to work differently compared with its expected behaviour Faults can be introduced in any level of software development They may come from design or coding, or external environment Depending upon the nature of faults, some faults in software can cause anything from a simple miscalculation to an entire system collapse. According to a survey detection and removal of software's faults cover around 50% 0f the total project budget therefore, finding and fixing faults as early as possible may save a lot of software development cost. It helps in identifying the fault proneness in software modules erstwhile testing phase using some underlying properties regarding the software system. Consequently, helps in allocating testing resources efficiently and carefully.

Abundant research has been done in the past to construct and evaluate the fault prediction models to predict the faulty modules in the software systems. Many statistical and machine learning techniques had been proposed and studied to predict the faults in a software system. These techniques include Logistic Regression, Neural Network, Naive Bayes, Discriminant Analysis, Classification Trees, Case-based Reasoning, Bagging, Boosting, and SVM. Despite the contribution made by them, there are still issues that prevent them from becoming widely adopted in practice. Most of the earlier software fault prediction studies have predicted the fault proneness of the software sub-modules in terms of faulty as well as non-faulty. There are quite a lot of disputes with this binary class classification. Even if the performance of the predicted models were reported excellent, the interpretation of the finding is hard to put into the proper usability context, idea and identification of the number of faults per module.

The subsequent insights of the assessment measures confirmed the prescient exactness and consistency of the manufactured fault expectation models. Software fault forecast plans to anticipate the fault inclined programming modules by utilizing some hidden properties of the product framework. It is planned to enable them to streamline the product quality confirmation endeavours and furthermore intended to advance the product fault distinguishing proof and expulsion endeavours and cost. Commonly, programming issue forecast is performed via preparing the expectation models utilizing venture properties loaded down with fault data for a known task and in this way, utilizing the prepared model to foresee shortcomings for the obscure project(s

All things considered in the event that we reuse the fabricated fault forecast demonstrate, it will re-raise a similar code zone as fault inclined. This is a general issue of fault forecast models, in the event that we utilize the code measurements. To deal with this issue, a few specialists have proposed diverse arrangement of measurements, for example, programming change measurements, record status measurements, and so forth. For building up a product fault forecast demonstrate, picking a superior learning calculation is appeared to be similarly imperative as choice of programming measurements and different parameters. Be that as it may, the choice of right fault expectation strategy is a testing issue. Various factors and factors impact this determination procedure. Already, a horde of various machine learning and factual strategies have been proposed and approved by different specialists to encourage fault forecast process. It incorporates the systems in light of Neural Network , Naive Bayes, Logistic Regression, Ensemble classifiers, Support Vector Machine and so on. There are varieties upon the utilization of these strategies to construct fault forecast show.

A few investigations built up the ease of use of a few methods for fault forecast, while different examinations inciting inquiries regarding a similar arrangement of procedures. One conceivable purpose behind this capricious execution of the fault expectation methods is that the vast majority of the investigations have utilized the fault forecast procedures as a black box without examining the space of the dataset.
As an exertion, this study displays a choice tree rationale (DTL) based suggestion framework that might be useful to the professionals or the scientists in the determination of the most suitable fault expectation procedure for a given target framework. In a product item,

in the event that we are foreseeing shortcomings over the diverse discharges, at that point a system substantial for one discharge ought to be reliably utilized for different discharges moreover.

The proposed framework at first recognized the fault dataset qualities of the given programming and the in view of these distinguished attributes the suggestion with respect to the appropriateness of fault expectation procedure is given. Thus, if there is no critical change in the fault attributes for a given discharge at that point there won't be change in the proposal with respect to fault forecast strategies. Yet, for the situation, there is a noteworthy change in the fault datasets qualities then the proposal with respect to the fault expectation procedures is changed.

As the world turned out to be increasingly reliant on innovation with each passing day, programming naturally turned into an essential organ for improvement. Since programming is required wherever today, its advancement is an exceedingly shrewd and exact process, including different advances. Known as programming advancement life cycle, these means incorporate arranging, examination, outline, improvement and usage, testing and upkeep. These means go ahead to make the ideal programming for customers. It's obvious that innovation is quickening at a fast pace and people are ending up assist subject to it for each reason.

1.1 Thesis Organization

This research work proposes an effective methodology using feature selection to identify the error rate of ten datasets using nine machine learning algorithms for fault prediction. AAE and ARE have been used to calculate the error rate. The feature selection allows to select important features that helps in improving the performance.

The rest of the thesis is as organized below: The first chapter provides a basic introduction of Fault prediction techniques and how faults in software modules affect software industry. The second chapter gives an account of the review of machine learning techniques and various technologies using which prediction have been done. The third chapter states the problem statement and objectives. This chapter deals with the main aim of carrying this research work and the objectives of the thesis. The fourth chapter deals with the tools and methodology used in the work. The fifth chapter identifies implementation and results regarding the thesis.Conclusion and Future scope is given in the sixth chapter.

CHAPTER-2
LITERATURE SURVEY

2.1 RELATED WORK

In the past few years there has been an emotional increment in chip away at Search Based Software Engineering, a way to deal with Software Engineering (SE) in which Search Based Optimization calculations are utilized to address issues in SE. SBSE has been connected to issues all through the SE life cycle, from prerequisites and task wanting to upkeep and reengineering. The approach is appealing on the grounds that it offers a suite of versatile mechanized and semi-robotized arrangements in circumstances encapsulated by substantial complex issue spaces with different contending and clashing targets. This study gives an audit and grouping of writing on SBSE. This research work recognizes look into patterns and connections between the methods connected and the applications to which they have been connected and features holes in the writing and roads for additionally investigate. Defect expectation on new undertakings or tasks with constrained authentic information is a fascinating issue in programming building. This is to a great extent since it is hard to gather surrender data to mark a dataset for preparing an expectation demonstrate. Cross-venture imperfection forecast (CPDP) has endeavoured to address this issue by reusing expectation models worked by different undertakings that have enough recorded information. Be that as it may, CPDP does not generally fabricate a solid forecast show due to the distinctive conveyances among datasets. Methodologies for imperfection expectation on unlabelled datasets have likewise attempted to address the issue by receiving unsupervised adapting however it has one noteworthy constraint, the need for manual exertion. In this examination, in propose novel methodologies, CLA and CLAMI that demonstrate the potential for deformity forecast on unlabelled datasets in a robotized way without requirement for manual exertion.

The key thought of the CLA and CLAMI approaches is to mark an unlabelled dataset by utilizing the size of metric qualities. Here exact investigation on seven open-source extends, the CLAMI approach prompted the promising forecast exhibitions, 0.636 and 0.723 in normal and AUC and f-measure that are tantamount to those of deformity expectation in light of administered learning.In 2002, the National Institute of Standards and Technology (NIST) evaluated that product absconds cost the U.S. economy in the zone of $60 billion multi year. It is notable that distinguishing and following these deformities productively measurably affects programming dependability. In this work, assess 104 scholarly papers on imperfection detailing distributed since the NIST answer to 2012 to recognize the most critical progressions in enhancing programming dependability however the effective ID and following of programming deserts. This paper arrange the examination into the regions of programmed deformity discovery, programmed imperfection settling, traits of deformity reports, nature of deformity reports, and triage of imperfection reports. In this paper at that point abridge the most critical work being done in every zone. At long last, in this paper give conclusions on the ebb and flow condition of the writing, propose apparatuses and exercises gained from the examination for training, and remark on open research problems.]During the product improvement process, forecast of the quantity of issues in programming modules can be more useful as opposed to anticipating the modules being defective or non-flawed. Such an approach may help in more engaged programming testing process and may upgrade the unwavering quality of the product framework. The vast majority of the prior chips away at programming flaw expectation have utilized arrangement methods for characterizing programming modules into defective or non-broken classes. The strategies, for example, Poisson relapse, negative binomial relapse, hereditary programming, choice tree relapse, and multilayer perceptron can be utilized for the expectation of the quantity of shortcomings.

In this paper, introduce a test concentrate to assess and analyse the ability of six fault expectation systems, for example, hereditary programming, multilayer perceptron, straight relapse, choice tree relapse, zero-swelled Poisson relapse, and negative binomial relapse for the forecast of number of issues. The test examination is completed for eighteen programming venture datasets gathered from the PROMISE information archive. The consequences of the examination are assessed utilizing normal outright blunder, normal relative mistake, measure of fulfilment, and forecast at level l measures. In this paper also perform Kruskal– Wallis test and Dunn's different correlation test to think about the relative execution of the considered fault forecast techniques.[5] Software deformity expectation assumes a critical part in evaluating the most imperfection inclined segments of programming, and an extensive number of studies have sought after enhancing expectation exactness inside an undertaking or crosswise over activities. In any case, the guidelines for settling on a suitable choice amongst inside and cross-venture imperfection expectations when accessible verifiable information are inadequate stay misty. The goal of this work is to approve the achievability of the indicator worked with a rearranged metric set for programming deformity forecast in various situations, and to research pragmatic rules for the decision of preparing information, classifier and metric subset of a given venture. To begin with, in light of six run of the mill classifiers, in this paper developed three kinds of indicators utilizing the measure of programming metric set in three situations. At that point, in this paper approved the worthy execution of the indicator in view of Top-k measurements as far as factual strategies. At last, in this paper approaches to limit the Top-k metric subset by expelling repetitive measurements, and this paper tried the solidness of such a base metric subset with one-way ANOVA tests. The test comes about to demonstrate that (1) the decision of preparing information ought to rely upon the particular prerequisite of forecast exactness; (2) the indicator worked with a rearranged metric set functions admirably and is exceptionally valuable in the event that constrained assets are provided; (3) straightforward classifiers (e.g., Naive Bayes) additionally have a tendency to perform well when utilizing a disentangled metric set for deformity expectation; and (4) in a few cases, the base metric subset can be distinguished to encourage the technique of general imperfection expectation with adequate loss of expectation accuracy by and by.

2.2 ALGORITHM USED

The Proposed work uses machine learning algorithms for fault prediction. The following sub-sections are categorized into various methods used for fault classifications.

2.2.1 RANDOM FOREST

Irregular Forest is an adaptable, simple to utilize machine learning calculation that produces, even without hyper-parameter tuning, an incredible outcome more often than not. It is likewise a standout amongst the most utilized calculations, since its effortlessness and the way that it can be utilized for both characterization and relapse assignments. In this post, you will realize how the irregular timberland calculation works and a few other vital things about it.

Arbitrary Forest is an administered learning calculation. Like you would already be able to see from its name, it makes a woods and makes it some way or another arbitrary. The forest it manufactures is a troupe of Decision Trees, more often than not prepared with the "sacking" strategy. The general thought of the packing strategy is that a blend of learning models expands the general outcome.

Random Forest pseudo code:

1. Randomly select "k" highlights from add up to "m" highlights. Where k << m

2. Among the "k" highlights, ascertain the hub "d" utilizing the best split point.

3. Split the hub into girl hubs utilizing the best split.

4. Repeat 1 to 3 stages until "l" number of hubs has been come to.

5. Build random forest by rehashing stages 1 upto 4 for "n" number circumstances to make "n" instances of trees. Initially, the algorithm starts by randomly selecting “Z” features out of total “X” features.

2.2.2 NEURAL NETWORK

Inside the field of machine learning, neural systems are a subset of calculations worked around a model of counterfeit neurons spread crosswise over at least three layers (we'll dive into the points of interest right away). There are a lot of other machine learning methods that don't depend on neural systems.

Inside neural systems, profound learning is by and large used to portray especially complex systems with numerous a greater number of layers than typical. The benefit of these additional layers is that the systems can grow considerably more noteworthy levels of deliberation, which is fundamental for certain mind boggling errands, similar to picture acknowledgment and programmed interpretation.

2.2.3 ADABOOST

AdaBoost Classifier

Boosting is an outfit method that endeavours to make a solid classifier from various feeble classifiers.

AdaBoost is best used to support the execution of choice trees on parallel characterization issues. AdaBoost was initially called AdaBoost.M1 by the creators of the system Freund and Schapire. All the more as of late it might be alluded to as discrete AdaBoost in light of the fact that it is utilized for characterization instead of relapse.

AdaBoost can be utilized to support the execution of any machine learning calculation. It is best utilized with frail students. These are models that accomplish exactness simply above irregular possibility on a grouping issue.

The best suited and thusly most basic calculation utilized alongside AdaBoost are choice trees with only single level. Since these trees are so tiny and just comprise of one choice for characterization, they are regularly called choice stumps.

Each example in the preparation dataset is weighted. The underlying weight is set to:

$$weight\ (xi) = 1/n$$

Where xi is the i'th preparing case and n is the quantity of preparing occasions.

2.2.4 Support Vector Machine

SVM is one of the regulated machine learning model which is for the most part utilized for classification and relapse investigation. SVM show breaks down information and perceives the examples associated with the informational index. SVM demonstrate goes about as a non-probabilistic twofold straight classifier by arranging input information into same classification or the other. SVM is by and large utilized for limiting the speculation mistake (genuine blunder) on inconspicuous case in light of Structural Risk Minimization rule. The fundamental type of SVM classifier, manages two-class issues, in which information are isolated by the ideal hyperplane defined by various help vectors. Bolster vectors are the subset of the preparation set which define the limit esteems between two classes. The general attributes of SVM are: Generalizes high dimensional spaces utilizing little preparing tests. Obtains worldwide ideal arrangement. Model non-straight useful relations. The principle objective of SVM is to outline a model which predicts target estimation of the dataset in the testing stage. Subsequently SVM goes about as a decent contender to plan a model in anticipating issue inclined modules. The general type of SVM work is defined as:

$$w * \varphi(x) + b$$

where φ(x) is non straight change. The fundamental objective of this examination is to ascertain the estimation of w and b, so the estimation of Y ′ can be found by minimization of relapse.Cost examination demonstrate Effectiveness of fault expectation Techniques hazard.

$$Rreg(Y') = C * lX\, i{=}0\, \gamma(Y'I - Yi) + 1\ 2 * kwk2$$

here γ speak to the cost function,constant esteem C speaks to punishments for estimation mistake (vast estimation of C implies that blunders are vigorously punished though a little estimation of C implies that mistakes are gently punished). A heavier punishment trains the relapse to limit mistakes by making less speculations. The estimation of w can be defined in type of information focuses as:

$$w = lXj=1(\alpha j - \alpha * j)\varphi(xj)$$

where α and $\alpha*$ speaks to the Lagrange multipliers , whose esteem is dependably more noteworthy and equivalent to zero i.e., $\alpha, \alpha* \geq 0$. So Equation 2.11 is modified as:

$$Y' = lXj=1\ (\alpha j - \alpha * j)\varphi(xj) * \varphi(x) + b = lXj=1\ (\alpha j - \alpha * j) * K(xj,x) + b$$

where G(xj,x) is the bit work, that empowers the speck item to be per shaped in high-dimensional component space utilizing low-dimensional space information. In writing direct, polynomial and spiral essential capacity utilized as a part. in this ponder polynomial capacity is utilized as portion function. Details For multiclass-characterization with g levels, g>2, lib svm utilizes the 'one to one'- approach, in which k(k-1)/2 parallel classifiers are prepared; the proper class is created by a voting plan. Libsvm inside utilizations an inadequate information portrayal, which is likewise abnormal state bolstered by the bundle SparseM. On the off chance that the indicator factors incorporate variables, the equation interface must be utilized to get a right model framework. plot.svm permits a basic graphical perception of grouping models. The likelihood demonstrate for grouping fits a calculated appropriation utilizing most extreme probability to the choice estimations of every double classifier, and figures the a-posteriori class probabilities for the multi-class issue utilizing quadratic streamlining. The probabilistic relapse display accept (zero-mean) laplace-conveyed mistakes for the expectations, and appraisals the scale parameter utilizing greatest probability.

2.2.5 Linear Regression

Relapse examination is a generally utilized measurable apparatus to build up a relationship demonstrate between two factors. One of these variable is called indicator variable whose esteem is accumulated through trials. The other variable is called reaction variable whose esteem is gotten from the indicator variable.

In Linear Regression these two factors are connected through a condition, where type (control) of both these factors is 1. Scientifically a direct relationship speaks to a straight line when plotted as a diagram. A non-direct relationship where the type of any factor isn't equivalent to 1 makes a bend.

Following is the depiction of the parameters utilized −

- y is the reaction variable.
- x is the indicator variable.
- a and b are constants which are known as the coefficients.

2.2.6 Bagging

Bootstrap Aggregation (Bagging)

Bootstrap Aggregation (or Bagging for short), is a basic and ground-breaking outfit technique.

A troupe strategy is a procedure that joins the forecasts from different machine learning calculations together to make more precise expectations than any individual model.

Bootstrap Aggregation is a general technique that can be utilized to decrease the change for those calculation that have high difference. A calculation that has high difference are choice trees, similar to order and relapse trees (CART).

Choice trees are touchy to the particular information on which they are prepared. In the event that the preparation information is changed (e.g. a tree is prepared on a subset of the preparation information) the subsequent choice tree can be very unique and thus the expectations can be very extraordinary.

Packing is the use of the Bootstrap strategy to a high-change machine learning calculation, commonly choice trees.

For instance, in the event that we had 5 packed away choice trees that made the accompanying class expectations for an in input test: blue, blue, red, blue and red, we would take the most successive class and foresee blue.

When packing with choice trees, we are less worried about individual trees overfitting the preparation information. Thus and for productivity, the individual choice trees are developed profound (e.g. hardly any preparation tests at each leaf-hub of the tree) and the trees are not pruned. These trees will have both high difference and low inclination. These are essential describe of sub-models when joining expectations utilizing stowing.

The main parameters when packing choice trees is the quantity of tests and thus the quantity of trees to incorporate. This can be picked by expanding the quantity of trees on pursue keep running until the point when the precision starts to quit indicating change (e.g. on a cross approval test bridle). Large quantities of models may set aside a long opportunity to get ready, yet won't overfit the preparation information.

CHAPTER 3
PROBLEM STATEMENT AND OBJECTIVES

This chapter states the problem in hand and lists out various objectives that are required to be met for solving the problem.

3.1 Problem Statement

As the software Industry is growing day by day, the software's are becoming more and more complex. These complex software's are not only difficult to debug but also makes a person scratch his head when classifying the modules on the basis of faulty or else non-faulty. The process of classification was used in early times of the software development. Here we have tried to use fault prediction in a module rather than classifying them into categories. We have used a number of algorithms to predict the number of faults in a particular dataset. Different fault prediction techniques have been used such as Linear Regression, random forest, SVM, Bagging, Boosting, MLP etc. The main goal is to discover the number of faults in particular modules with high rate of prediction so that debugging could be narrowed down in future.

3.2 Thesis Objectives

Various objectives that are needed to be fulfilled to solve the problem in hand are listed as below:

- To study various techniques and tools available for finding which software modules are frequently faulty.
- To find out the error rate using the predicted and actual values using the fault prediction techniques.
- To use feature selection to use most important metrics of softwares.
- To develop more efficient machine learning algorithm that could boost the process of finding the faults more efficiently using different datasets.

CHAPTER 4
TOOLS AND METHODOLOGY

4.1 Tools

Tools that are used for implementation of the problem solution are as follows:

- R Studio: Version 0.99.473 - © 2009-2015 R Studio, Inc.
- Microsoft Excel 2013

4.2 Methodology

The methodology is used for building multiple classification models. The model methods are categorized into 2 sub-modules. The first module has been used to predict the error rates namely AAE and ARE for all ten datasets. Also, the Kruskal-wallis test has been conducted to find the significant difference in the performance. The second sub module describes the selection of important features using Boruta. Different performace evaluation metrics have used to determine the best prediction model. Various classification models have been studied.

4.2.1 Using R Tool on Standalone machine Environment

The R computer programs are an essential tool for progression in the numeric examination and machine learning spaces. R is a perfect way to deal with make reproducible, extraordinary examination. R is extensible and offers rich value for architects to manufacture their own specific gadgets and procedures for examining data. With machines winding up recognizably more basic as data generators, the noticeable quality of the dialects must be depended upon to create. When it at first turned out, the best-favored angle was that it was free programming. The vastness of package organic framework is irrefutably one of the R's most grounded qualities - if a true technique exists, odds are there's presently an R package out there for it. R's positive conditions fuse its package natural framework.

Here, the accuracy of different machine learning algorithms has been explored using R Tool on the Standalone machine. Here initial analysis has been done using Microsoft excel. A csv file has been provided as an input for R-Studio. Analysis has been done using programming language R.

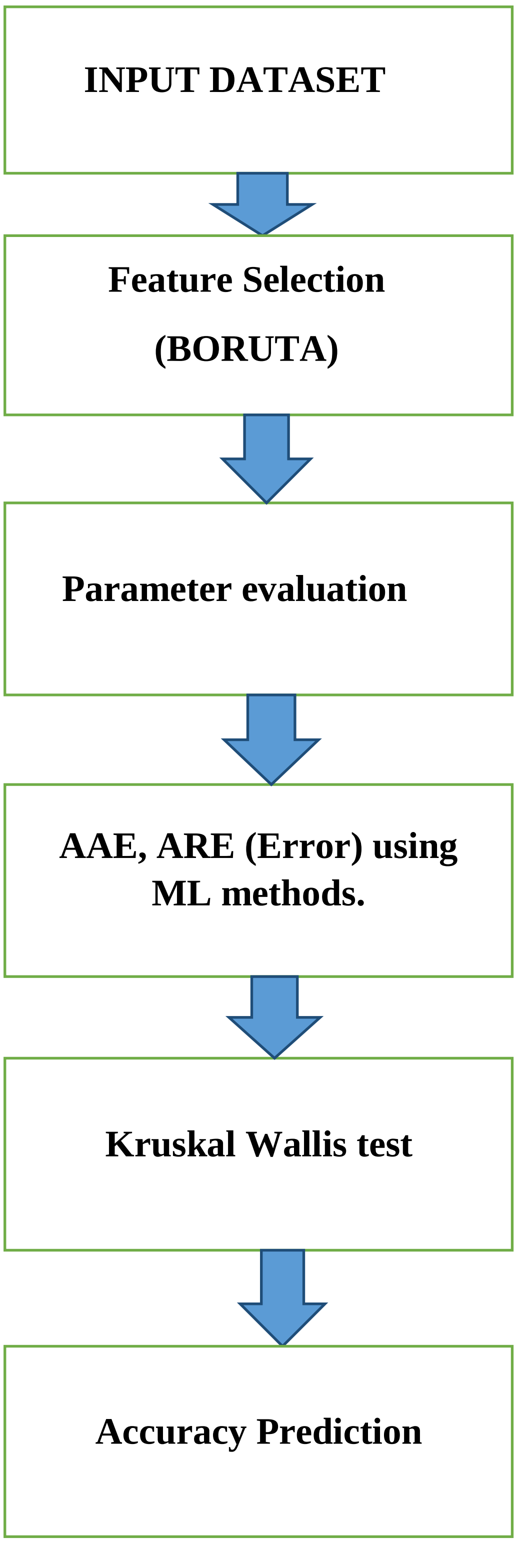


Fig. 4.1 Block Diagram of Proposed work

- The data is gathered from web sources after which Pre-processing of Data is done which includes Data cleaning, Integration of data, and Data Transformation.
- Different evaluation measures are used. Henceforth, the outcomes have been analyzed on the basis of error rate and accuracy.

- In any case, R has both upsides and downsides that designers ought to know. With enthusiasm for the programming developing, as appeared on language notoriety files, for example, TIobe, Redmond and PyPL[5], R initially showed up in the 1990s and has filled in as an execution of the S measurable programming languages.

- "R is the most mainstream dialect utilized as a part of the field of statistics." It has all the adaptability and power. R is in reality only accumulations of scripts that are sorted out into projects."
- Data purifying/cleaning is a term identified with getting the significant data from the crude information and noisy data removal (information not profitable to us). This should be possible effectively in Microsoft Excel and is a generally utilized strategy for each information researcher.

4.3 Evaluation Criteria Used for Classification

4.3.1 Correlation Matrix

The confusion matrix is also called as Error matrix. It is a table that is often used to describe the performance of a classification method on a set of test data for which actual value are known. Each tuple of the matrix denotes the occurrences in the actual class. Each column of the matrix denotes the occurances in a predicted class. The correlation matrix is represented as:

		Predicted	
		No	**Yes**
Actual	**No**	**TN**	**FP**
	Yes	**FN**	**TP**

True Positive: Case in which a Fault predicted is "Yes" and the Fault actually exist.

True Negative: Case in which a Fault predicted "No" and the Fault actually don't exist.

False Positive: Case in which a Fault predicted "Yes" and the Fault actually don't exist.

False Negative: Case in which a Fault predicted “No”, but the Fault actually exist.

4.3.2 Accuracy and Precision

In classification, accuracy and precision are two important evaluation parameters. Accuracy is defined as the sum total of true positive and true negative instances divided by 100.And Precision is fraction of true positive and predicted yes instances. The formula of Accuracy and Precision are given below:

Accuracy: $\frac{\text{TP+TN}}{100}$ Precision: $\frac{\text{TP}}{\text{Predicted Yes}}$

4.3.3 Recall and F-Square

Recall is defined as the fraction between True Positive instances and Actual yes instances whereas F-Square is the fraction between product of the recall and precision to the summation of recall and precision parameter of classification. The formula of recall and precision given below:

Recall: $\frac{\text{TP}}{\text{Actual Yes}}$ F-Square: $\frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$

4.3.4 Sensitivity, Specificity and ROC

Sensitivity is defined as the fraction of true positive and actual yes instances whereas specificity is the difference between one and false positive rate value. ROC is defined as the fraction between the true positive rate and the false positive rate.

Sensitivity: $\frac{\text{TP}}{\text{Actual Yes}}$ Specificity: $1- \frac{\text{FP}}{\text{Actual No}}$

ROC: $\frac{\text{TPR}}{\text{FPR}}$

4.3.4 MCC

MCC is a measure that studies both true and false positives and negatives. The MCC can be obtained as

$$MCC = \frac{(TP)(TN) - (FP)(FN)}{\sqrt{[TP + FP][TP + FN][TN + FP][TN + FN]}}$$

CHAPTER-5
IMPLEMENTATION AND RESULTS

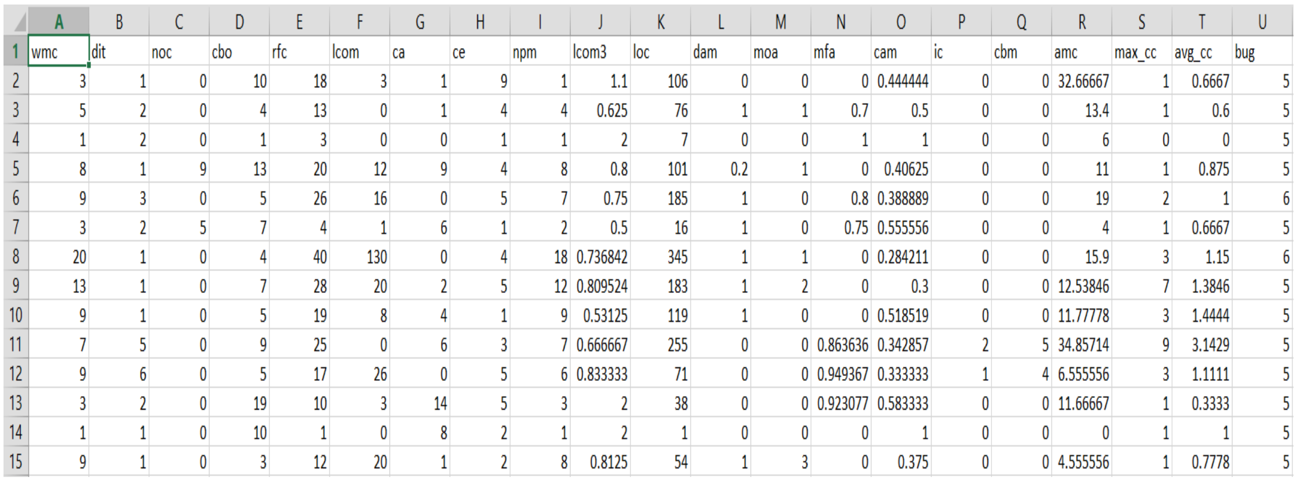
This chapter incorporates an investigation that is performed when proposed technique is applied on dataset regarding destinations that are accomplished and weaknesses of different existing methodologies. Relative examination demonstrates that proposed strategy gives better outcomes in term of the outcome produced by existing model. Experimental results have been described in tabular form. This chapter is divided into 3 sections:

1. Measuring performance using R Tool on standalone machine.
2. Comparison between the results

5.1 Measuring performance using R Tool on standalone machine

Implementation technique using R Tool on standalone machine can be illustrated in following flow of data:

- Dataset collection and description
- Data Cleansing Phase
- Training and Testing Data
- Models Applied for Predictions
- Result: Final Prediction



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	max_cc	avg_cc	bug
2	3	1	0	10	18	3	1	9	1	1.1	106	0	0	0	0.444444	0	0	32.66667	1	0.6667	5
3	5	2	0	4	13	0	1	4	4	0.625	76	1	1	0.7	0.5	0	0	13.4	1	0.6	5
4	1	2	0	1	3	0	0	1	1	2	7	0	0	1	1	0	0	6	0	0	5
5	8	1	9	13	20	12	9	4	8	0.8	101	0.2	1	0	0.40625	0	0	11	1	0.875	5
6	9	3	0	5	26	16	0	5	7	0.75	185	1	0	0.8	0.388889	0	0	19	2	1	6
7	3	2	5	7	4	1	6	1	2	0.5	16	1	0	0.75	0.555556	0	0	4	1	0.6667	5
8	20	1	0	4	40	130	0	4	18	0.736842	345	1	1	0	0.284211	0	0	15.9	3	1.15	6
9	13	1	0	7	28	20	2	5	12	0.809524	183	1	2	0	0.3	0	0	12.53846	7	1.3846	5
10	9	1	0	5	19	8	4	1	9	0.53125	119	1	0	0	0.518519	0	0	11.77778	3	1.4444	5
11	7	5	0	9	25	0	6	3	7	0.666667	255	0	0	0.863636	0.342857	2	5	34.85714	9	3.1429	5
12	9	6	0	5	17	26	0	5	6	0.833333	71	0	0	0.949367	0.333333	1	4	6.555556	3	1.1111	5
13	3	2	0	19	10	3	14	5	3	2	38	0	0	0.923077	0.583333	0	0	11.66667	1	0.3333	5
14	1	1	0	10	1	0	8	2	1	2	1	0	0	0	1	0	0	0	1	1	5
15	9	1	0	3	12	20	1	2	8	0.8125	54	1	3	0	0.375	0	0	4.555556	1	0.7778	5

5.1.1 Performance evaluation measures

Distinctive execution assessment measures are utilized to assess the forecast comes about. These execution assessment measures are normal total mistake, measure of culmination , normal relative blunder, and expectation at level. The portrayal of these methods is given in this area.

(I) Average total mistake (AAE): AAE measures the normal extent of the blunders in an arrangement of expectation. It demonstrates the contrast between the anticipated esteem and the genuine esteem. It is defined by Eq. 1

$$AAE = \left(1/n\right)\sum_{i=1}^{n} |\left(\bar{y_i} - y_i\right)|$$

(ii) Average relative blunder (ARE): ARE figures how huge the outright mistake is contrasted and the aggregate size of the protest estimated. It is defined by Eq. 1.

$$ARE = (1/n)\sum_{i=1}^{n} |(\bar{y_i} - y_i)| / (y_i + 1)$$

Here, Yi is the anticipated number of shortcomings in a product module and Yi is the relating real estimation of flaws. N is the quantity of modules. On account of ARE, now and again estimation of Yi can be zero. To alleviate this issue, we included '1' with the estimation of Yi at the denominator to make the definition constantly very much defined (Khoshgoftaar et al. 1992). A little estimation of AAE and ARE measures demonstrates that we have a decent expectation display (Conte et al. 1986).

Where k is the quantity of perceptions whose esteem exists in l% of the genuine qualities, l is the limit esteem, and n is the aggregate number of perceptions. R. (Veryard 2014) recommended that for a model to be viewed as satisfactory, estimation of l ought to be not exactly or equivalent to 0.30. Accordingly, we have utilized 0.30 as the estimation of l. This implies the anticipated esteem must be inside the 30% scope of the genuine incentive to consider the forecast satisfactory for the given module. Measure of culmination: Completeness of a forecast demonstrate is defined as a proportion of the quantity of shortcomings found in the modules anticipated as defective to the aggregate number of deficiencies in the product framework (Briand and Jurgen 2002). It demonstrates the level of flaws that have been discovered when we utilized the given forecast show.

5.2 **RESULTS**

Initially, we examine the results of AAE and ARE for the fault prediction techniques used in the proposed work. Furthermore, we discuss the results of prediction using feature selection and Kruskal–Wallis test.

5.2.1 **AAE and ARE analysis**

The analysed result of AAE and ARE for ten datasets are shown in table 5.1 shows. The table includes the median values for both :relative error and absolute error for the ten used datasets. The comparison based on the ARE and AAE values shows that the Adaboost and bagging techniques generated the lowest error values in most of the datasets and displayed better prediction accuracy as compared to the other analysed fault prediction techniques. Boosting and Linear regression are the 3rd and 4th best fault prediction techniques. Multilayer perceptron (MLP) stood last with lowest prediction accuracy, the AAE and ARE values of MLP and Random forest were higher than compared to other used fault prediction techniques.

The ARE and AAE for various datasets used with different algorithms have been shown in Table 5.1. ARE and AAE are calculated using the predicted and actual values from the model's prediction. Nine models have been applied on ten datasets and different error rates have been calculated using R studio.

Table 5.1 AAE and ARE values of nine fault prediction techniques for all datasets.

Datasets		**Neural network**	**Bagging**	**Random forest**	**Boosting**
ant 1.7	AAE	0.14	**0.08**	0.24	**1.52**
	ARE	0.02	**0.01**	0.15	**0.95**
camel 1.2	AAE	0.68	**0.15**	0.33	0.62
	ARE	0.40	**0.11**	0.27	0.40
camel 1.4	AAE	0.34	0.12	0.24	0.56
	ARE	0.24	**0.04**	0.11	0.35
camel 1.6	AAE	0.15	0.21	0.15	0.24
	ARE	0.11	0.15	0.06	0.05
Xalan 2.4	AAE	0.26	**0.10**	0.20	**0.50**
	ARE	0.22	**0.05**	0.16	**0.42**
Xalan 2.5	AAE	0.35	**0.22**	**1.24**	0.35
	ARE	0.27	0.12	**0.51**	**0.11**
Xalan 2.6	AAE	0.34	0.25	0.60	0.34
	ARE	0.20	0.14	**0.08**	0.21
xerces-1.3	AAE	0.39	0.16	0.41	0.60

	ARE	0.21	0.03	0.13	**0.40**
xerces-1.4	AAE	0.44	0.56	0.63	0.29
	ARE	0.16	0.30	0.34	0.14
Prop V4	AAE	**0.14**	**1.25**	0.29	0.22
	ARE	0.06	**0.93**	0.13	0.09

Datasets		**Linear Regression**	**decision tree**	**MLP**	**Adaboost**	**SVM**
ant 1.7	AAE	0.35	0.36	0.56	0.09	0.39
	ARE	0.21	0.21	0.24	0.01	0.20
camel 1.2	AAE	1.15	**1.21**	0.75	0.19	0.30
	ARE	**0.88**	0.50	0.43	0.12	0.19
camel 1.4	AAE	0.45	0.44	**0.59**	**0.09**	0.42
	ARE	0.31	0.29	**0.37**	0.06	0.24
camel 1.6	AAE	0.60	0.42	**0.96**	**0.11**	0.50
	ARE	0.38	0.42	**0.67**	0.06	0.19
Xalan 2.4	AAE	0.24	0.21	0.26	0.12	0.21
	ARE	0.17	0.19	0.20	0.08	0.13
Xalan 2.5	AAE	0.57	0.45	0.64	0.28	0.56
	ARE	0.36	0.29	0.44	0.22	0.35
Xalan 2.6	AAE	0.47	0.40	**0.61**	**0.19**	0.52
	ARE	0.32	0.26	**0.36**	0.13	0.32
xerces-1.3	AAE	0.56	0.33	**0.70**	**0.09**	0.42
	ARE	0.39	0.20	0.48	0.06	0.19
xerces-1.4	AAE	1.75	1.59	**2.11**	0.06	0.63
	ARE	**0.56**	0.42	0.66	0.16	**0.07**
ProP v4	AAE	0.23	0.18	0.15	0.19	0.23
	ARE	0.17	0.12	0.09	**0.09**	0.20

The Box-plot of both AAE and ARE measures have been shown in Fig. 5.1 and Fig. 5.2. X-axis specifies the fault prediction techniques that have been used, and y-axis specifies the median values of the errors that were produced by the fault prediction techniques.

While evaluating the AAE measure, RF and MLP techniques produced the lowest median value, DT and SVM produced moderate median values, whereas NN produced the highest median values. Boosting generates the lowest minimum values, LR produced moderate minimum values, and MLP and RF produced the highest minimum values. For maximum value, LR and Boosting produced the minimum values, MLP and LR produced moderate values, RF and DT generates the highest values.

With respect to ARE measure, SVM and LR generated the lowest median values, Boosting and RF generates moderate median values, and DTR and MLP produced the highest median values. Similarly, Adaboost and Bagging produced the lowest minimum values, RF and NN produced moderate minimum values, and SVM and LR produced the highest minimum values. Adaboost generated the lowest maximum value, LR, Boosting, and MLP produced the moderate maximum values, and DTR and MLP produced the highest maximum values.

Overall, it is clear from the figures that for both the measures (AAE and ARE), the performance of Adaboost and Bagging were almost similar than other fault prediction techniques. RF and SVM predicts moderate results , while the predictions of MLP and Bosting poor compared to other fault prediction techniques we have used.

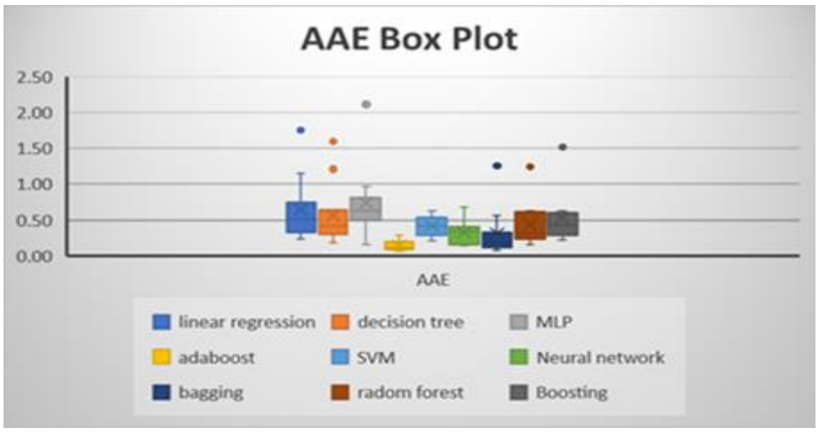


Fig 5.1: Boxplot for AAE

Various algorithms and their respective AAE’s are shown in Fig. 5.1. As it can be seen clearly that MLP has the highest error rate and adaboost performed well with lowest error rate.

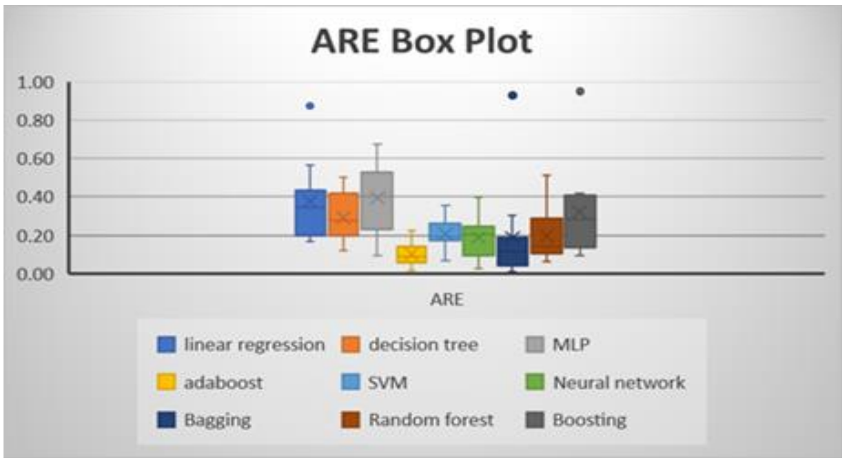


Fig 5.2: Boxplot for ARE

a)

K(observed value)	207.927
K(Critical value)	11.07
DF	5
P value(two tailed)	<0.0001
Alpha	0.05

Table 5.2 Representation of Kruskal-wallis test results over ARE.

H0: H0 is the samples from the same group of population. Ha: The samples are not from the same group of population. Here the calculated *p* value is less than the significance level alpha=0.05, the null hypothesis H0 is rejected and the alternative hypothesis Ha is accepted.

b)

K (observed value)	157.269
K (Critical value)	11.07
DF	5
P value (two tailed)	<0.0001
Alpha	0.05

Table 5.3 Representation of Kruskal-wallis test results over AAE.

H0: H0 is the samples from the same group of population. Ha: The samples are not from the same group population. Here, the calculated *p* value is less than the significance level alpha=0.05. so, the null hypothesis H0 is rejected and H1 is accepted as the alternative hypothesis.

Kuskal-Wallis test is conducted here to calculate the difference in the performance statistically.

Kruskal–Wallis is considered to be a nonparametric test that tests whether the set of samples are taken from the same set of population or from some other. Relative error and Absolute error values of each used dataset are used to perform the Kruskal Wallis test. The results after applying Kruskal–Wallis test are shown in Tables 5.2 and 5.3 The results of the test indicate that the nine fault prediction techniques which have been used in this study have statistically substantial performance difference from one another for at least one sample. During the Kruskal wallis test, the, i.e., α(significant value) is set as 0.05, which represents 95% of the confidence interval. p value is larger than the α value for both the Average relative error and Average absolute error , thereby accepting the alternate hypothesis. which represents that the nine fault prediction techniques used in the study generated results that were suggestively different as compared to the other used techniques.

– While considering the AAE measure, the treatments that included Adaboost and Boosting techniques have performed noticeably different from the remaining fault prediction techniques. While the remaining treatments showed no significant changes at all.

–While considering the ARE measure, the treatments that included Adaboost, DTR, and Bagging techniques showed results that performed different from the other used fault prediction techniques. For other treatments, LR has shown a bit of difference, other than that no significant difference has been found.

– According to our overall findings Adaboost, SVM, NN fault prediction techniques did perform almost same as each other, whereas NN, DTR and MLP fault prediction techniques performed quite a bit different from the other techniques used for fault prediction.

5.3 Feature Selection using Boruta

Feature selection techniques are used to select features that tend to improve the accuracy of the particular algorithm on a dataset. Here, we have used machine learning based techniques for example Linear regression, decision tree, bagging, boosting, multilayer perceptron, support vector machine, Adaboost are being used to predict the faulty and defective and non- defective modules. Boruta chooses the most critical element among every one of the features from the every other dataset.

Moreover, Boruta here is used to choose 10 number of features, 16 number of features, 20 number of features among every one of the features. Here, the execution of different machine learning based techniques, for example, bagging, boosting, random forest, decision tree, Adaboost, support vector machine, multilayer perceptron , Linear regression based strategies are processed with the distinctive diverse highlights chose by Boruta selection algorithm.

Here, it is clearly visible that the Neural Network with 10 features has the accuracy of 94.49% with ROC as 0.969, Precision 0.955 and MCC equal to 0.8791.

When 16 features were used, Adaboost performed well as compared to all other machine learning algorithms with the accuracy of 91.53, ROC as 0.939, precision as 0.933 and MCC as 0.7723.

As far as 20 features are concerned Linear model predicted the highest accuracy with 91.53 % , ROC as 0.939, Precision as 0.933 and MCC as 0.7723019.

Table 5.4 represents the prediction measures of all the datasets namely ROC, Precision, Accuracy, MCC using 10 features of the dataset that were selected by Boruta.

5.3.1 10-Features selected by Boruta

10 Features selected by Boruta						
S.No.	**CLASSIFIRES**	**DATASET NAME**	**ROC**	**PRECISION**	**ACCURACY**	**MCC**
1	AdaBoost	ant1.7	0.789	0.632	79.89	0.3617787
		camel1.4	0.7	0.357	80.23	0.0897396
		camel1.6	0.673	0.303	78.29	0.0943209
		camel1.2	0.565	0.448	63.11	0.0703262
		prop-4	**0.766**	**0.568**	**90.97**	**0.2402734**
		xalan-2.4	0.812	0.375	84.14	0.2065083
		xalan-2.5	0.68	0.656	61.8	0.243765

		xalan-2.6	0.817	0.791	75.21	0.5064628
		xerces-1.3	0.762	0.6	86.81	0.3808127
		xerces-1.4	0.93	0.916	90.68	0.760148
2	Bagging	ant1.7	0.734	0.413	79.6	0.2632706
		camel1.4	0.5	0.203	79.66	0
		camel1.6	0.563	0.481	76.49	0.2348043
		camel1.2	0.514	0.462	65.16	0.0585552
		prop-4	0.578	0.588	91.14	0.2420488
		xalan-2.4	0.701	0.5	83.79	0.33419
		xalan-2.5	0.601	0.529	58.07	0.2339586
		xalan-2.6	0.762	0.782	73.8	0.4750398
		xerces-1.3	0.636	0.625	84.62	0.3663305
		xerces-1.4	**0.893**	**0.953**	**92.37**	**0.8072525**
3	decision tree	ant1.7	0.642	0.519	78.28	0.3698361
		camel1.4	0.523	0.357	83.07	0.0976947
		camel1.6	0.532	0.294	77.43	0.0996254
		camel1.2	0.519	0.397	60.98	0.0464367
		prop-4	**0.59**	**0.557**	**90.87**	**0.2684842**
		xalan-2.4	0.624	0.341	81.77	0.1971909
		xalan-2.5	0.628	0.58	62.19	0.2655879
		xalan-2.6	0.751	0.726	70.65	0.4138944
		xerces-1.3	0.592	0.5	87.67	0.2706026
		xerces-1.4	0.922	0.944	91.53	0.7830603

4	linear model	ant1.7	0.817	0.641	80.94	0.3821535
		camel1.4	0.623	0.5	80.52	0.1018991
		camel1.6	0.656	0.462	79.07	0.1173185
		camel1.2	0.506	0.4	60.25	0.0706919
		prop-4	**0.726**	**0.659**	**89.88**	**0.1970113**
		xalan-2.4	0.717	0.619	86.21	0.3579673
		xalan-2.5	0.602	0.633	58.39	0.1814522
		xalan-2.6	0.694	0.629	61.97	0.2427418
		xerces-1.3	0.532	0.238	76.37	0.0590791
		xerces-1.4	0.947	0.954	89.83	0.7291573
5	neural network	ant1.7	0.83	0.237	80.27	0.074324
		camel1.4	0.529	0.175	46.99	0.0284271
		camel1.6	0.657	0.714	78.04	0.1606412
		camel1.2	0.581	0.465	70.79	0.1531139
		prop-4	0.761	0.412	89.91	0.1037427
		xalan-2.4	0.601	0.2	66.55	0.1137807
		xalan-2.5	0.649	0.592	57.45	0.1570248
		xalan-2.6	0.694	0.629	61.97	0.2427418
		xerces-1.3	0.53	0.129	59.49	-0.035968
		xerces-1.4	**0.969**	**0.955**	**94.49**	**0.8791806**
6	Svm	ant1.7	0.767	0.228	80.97	0.428279
		camel1.4	0.692	0.634	83.75	0.679524
		camel1.6	0.625	0.286	79.92	0.0279036

		camel1.2	0.607	0.583	61.89	0.0904813
		prop-4	0.651	0.441	**90.02**	0.1139886
		xalan-2.4	0.652	0.167	82.76	0.0032012
		xalan-2.5	0.686	0.621	62.42	0.245856
		xalan-2.6	0.765	0.82	72.96	0.4769941
		xerces-1.3	0.791	0.714	86.26	0.3106639
		xerces-1.4	0.861	0.849	84.75	0.598613
7.	MLP	ant1.7	0.762	0.782	73.8	0.4750398
		camel1.4	0.636	0.625	84.62	0.3663305
		camel1.6	0.625	0.286	79.92	0.0279036
		camel1.2	0.607	0.583	61.89	0.0904813
		prop-4	0.514	0.462	65.16	0.0585552
		xalan-2.4	**0.578**	**0.588**	**91.14**	**0.2420488**
		xalan-2.5	0.673	0.303	78.29	0.0943209
		xalan-2.6	0.565	0.448	63.11	0.0703262
		xerces-1.3	0.341	0.523	81.77	0.1971909
		xerces-1.4	0.58	0.519	62.19	0.2655879
8.	Boosting	ant1.7	0.642	0.519	78.28	0.3698361
		camel1.4	0.743	0.357	80.23	0.0897396
		camel1.6	0.656	0.462	79.07	0.1173185
		camel1.2	0.557	0.564	62.73	0.15334
		prop-4	**0.627**	**0.621**	**89.91**	**0.2332957**
		xalan-2.4	0.734	0.444	85.52	0.2286648

		xalan-2.5	0.781	0.432	83.79	0.630754
		xalan-2.6	0.672	0.598	62.30	0.0178312
		xerces-1.3	0.592	0.55	77.67	0.2706026
		xerces-1.4	0.916	0.898	83.05	0.5808155
9.	Random forest	ant1.7	0.93	0.537	85.22	0.6442091
		camel1.4	0.502	0.204	79.37	0.0270905
		camel1.6	0.623	0.2623	84.82	0.2865209
		camel1.2	0.706	0.901	83.93	0.0655102
		prop-4	**0.693**	**0.165**	**93.70**	**-0.192982**
		xalan-2.4	0.519	0.491	87.76	0.2071909
		xalan-2.5	0.749	0.95	61.19	0.2451879
		xalan-2.6	0.773	0.576	73.17	0.4385006
		xerces-1.3	0.674	0.292	42.60	-0.027488
		xerces-1.4	0.407	0.672	45.23	0.0253398

Table 5.4 10-Features selected by Boruta

Table 5.4 represents the prediction measures of all the datasets namely ROC, Precision, Accuracy, MCC using 16 features of the dataset that were selected by Boruta.

5.4 16-Features Selected

16 Features selected by Boruta						
S.No.	**CLASSIFIRES**	**DATASET NAME**	**ROC**	**PRECISION**	**ACCURACY**	**MCC**
1	AdaBoost	ant1.7	0.76	0.622	80.08	0.372531487
		camel1.4	0.732	0.625	83.95	0.1887846
		camel1.6	0.691	0.417	78.04	0.168867

		camel1.2	0.598	0.462	64.34	0.1076602
		prop-4	0.792	0.58	90.71	0.2331917
		xalan-2.4	0.757	0.522	83.1	0.2714224
		xalan-2.5	0.643	0.621	61.18	0.2240989
		xalan-2.6	0.828	0.777	73.24	0.4667699
		xerces-1.3	0.796	0.5	84.07	0.4135098
		xerces-1.4	**0.939**	**0.933**	**91.53**	**0.7723019**
2	Bagging	ant1.7	0.64	0.509	77.59	0.312280179
		camel1.4	0.502	0.204	79.37	0.0270905
		camel1.6	0.502	0.204	79.37	0.0270905
		camel1.2	0.521	0.625	66.8	0.1106934
		prop-4	**0.569**	**0.481**	**90.68**	**0.2014914**
		xalan-2.4	0.583	0.571	86.21	0.2730102
		xalan-2.5	0.598	0.552	59.32	0.1968034
		xalan-2.6	0.728	0.709	69.58	0.3918305
		xerces-1.3	0.707	0.812	89.01	0.5419325
		xerces-1.4	0.933	0.913	89.83	0.7515565
3	decision tree	ant1.7	0.72	0.588	79.89	0.440526974
		camel1.4	0.52	0.357	81.69	0.087945
		camel1.6	0.557	0.397	77.02	0.1982736
		camel1.2	0.523	0.38	64.26	0.0589195
		prop-4	0.57	0.59	90.3	0.1466443
		xalan-2.4	0.525	0.75	85.08	0.1740127

		xalan-2.5	0.598	0.552	59.32	0.1968034
		xalan-2.6	0.763	0.723	73.36	0.466483
		xerces-1.3	0.608	0.643	85.46	0.3330722
		xerces-1.4	**0.837**	**0.917**	**89.15**	**0.7238917**
4	linear model	ant1.7	0.76	0.595	80.64	0.343662599
		camel1.4	0.678	0.444	80.82	0.1060837
		camel1.6	0.698	0.467	80.36	0.1386214
		camel1.2	0.557	0.5	62.73	0.15334
		prop-4	**0.727**	**0.621**	**90.91**	**0.2332957**
		xalan-2.4	0.704	0.444	85.52	0.2286648
		xalan-2.5	0.64	0.617	59.94	0.2007405
		xalan-2.6	0.774	0.772	72.39	0.4490148
		xerces-1.3	0.736	0.5	86.26	0.327062
		xerces-1.4	0.905	0.878	87.29	0.6626029
5	neural network	ant1.7	0.83	0.221	83.61	0.5264874
		camel1.4	0.504	0.144	72.74	-0.0397308
		camel1.6	0.566	0.237	72.87	0.0313058
		camel1.2	0.462	0.195	65.8	0.0432402
		prop-4	0.548	0.11	55.73	0.0429783
		xalan-2.4	**0.576**	**0.161**	**89.97**	**0.0775333**
		xalan-2.5	0.619	0.57	23.91	0.2012869
		xalan-2.6	0.78	0.76	71.27	0.4569006
		xerces-1.3	0.574	0.132	45.6	-0.0252882

		xerces-1.4	0.517	0.772	40.25	0.0653398
6	Svm	ant1.7	0.75	0.233	78.82	0.8402591
		camel1.4	0.63	1	83.07	0.1052157
		camel1.6	0.666	0.375	80.54	0.0609839
		camel1.2	0.622	0.625	62.7	0.1280282
		prop-4	**0.673**	**0.56**	**90.71**	**0.1359105**
		xalan-2.4	0.781	0.84	83.79	0.630754
		xalan-2.5	0.74	0.72	62.11	0.2730518
		xalan-2.6	0.797	0.797	72.68	0.4611323
		xerces-1.3	0.781	0.714	87.36	0.3265986
		xerces-1.4	0.916	0.898	83.05	0.5808155
7.	Random forest	ant1.7	0.458	0.461	73.51	0.4652368
		camel1.4	0.623	0.2623	84.82	0.2865209
		camel1.6	0.556	0.583	79.92	0.2104191
		camel1.2	0.522	0.7	63.28	0.1212366
		prop-4	0.729	0.636	82.43	0.3582531
		xalan-2.4	0.531	0.571	84.81	0.1618251
		xalan-2.5	0.601	0.569	60.2	0.2024827
		xalan-2.6	0.703	0.655	69.98	0.4046577
		xerces-1.3	0.508	0.25	85.46	0.0442431
		xerces-1.4	**0.844**	**0.972**	**91.23**	**0.8067342**
8.	MLP	camel1.4	0.745	0.634	74.21	0.0615025
		camel1.6	0.738	0.592	81.64	0.1700923
		camel1.2	0.696	0.201	63.2	-0.0458102

		prop-4	**0.596**	**0.265**	**92.70**	**-0.1739102**
		xalan-2.4	0.829	1.304	84.83	0.2937467
		xalan-2.5	0.859	1.299	73.16	0.2937467
		xalan-2.6	0.672	0.598	62.30	0.0178312
		xerces-1.3	0.507	0.636	60.15	0.0178312
		xerces-1.4	0.491	0.749	70.89	0.0097906
		camel1.4	0.763	0.537	74.21	0.0636155
9.	Boosting	ant1.7	0.722	0.519	78.28	0.3698361
		camel1.4	0.523	0.357	83.07	0.0976947
		camel1.6	0.532	0.294	77.43	0.0996254
		camel1.2	0.519	0.397	60.98	0.0464367
		prop-4	**0.59**	**0.557**	**90.61**	**0.2684842**
		xalan-2.4	0.624	0.341	81.37	0.1971909
		xalan-2.5	0.628	0.86	61.19	0.2655879
		xalan-2.6	0.751	0.726	73.32	0.4138944
		xerces-1.3	0.592	0.55	77.67	0.2706026
		xerces-1.4	0.922	0.944	91.53	0.7830603

Table 5.5 16-Features selected by Boruta

Table 5.6 represents the prediction measures of all the datasets namely ROC, Precision, Accuracy, MCC using 20 features of the dataset that were selected by Boruta.

5.5 20-Features selected

RESULT WITH 20 FEATURES

S.No.	CLASSIFIRES	DATASET NAME	ROC	PRECISION	ACCURACY	MCC
1	AdaBoost	ant1.7	0.759	0.5	78.93	0.349548496
		camel1.4	0.683	0.3	80.8	0.053356
		camel1.6	0.701	0.647	82.17	0.2484638
		camel1.2	0.627	0.708	70.49	0.2603006
		prop-4	0.845	0.773	91.83	0.3866511
		xalan-2.4	0.791	0.45	83.79	0.2216205
		xalan-2.5	0.66	0.612	62.11	0.2400709
		xalan-2.6	0.772	0.699	70.42	0.4049712
		xerces-1.3	0.836	0.55	87.91	0.4342182
		xerces-1.4	**0.948**	**0.947**	**93.64**	**0.8434175**
2	Bagging	ant1.7	0.729	0.534	80.27	0.389494174
		camel1.4	0.515	0.375	81.66	0.0790847
		camel1.6	0.513	0.375	80.88	0.0705456
		camel1.2	0.516	0.667	64.34	0.0995858
		prop-4	0.729	0.728	90.97	0.3398778
		xalan-2.4	0.547	1	86.55	0.2834401
		xalan-2.5	0.618	0.583	61.49	0.24482
		xalan-2.6	0.699	0.651	69.58	0.399011
		xerces-1.3	0.658	0.562	87.36	0.3834306
		xerces-1.4	**0.884**	**0.935**	**91.43**	**0.8027267**
3	decision tree	ant1.7	0.778	0.461	78.55	0.332514652
		camel1.4	0.504	0.2	83.52	0.0209286

		camel1.6	0.556	0.583	79.92	0.2104191
		camel1.2	0.522	0.7	63.28	0.1212366
		prop-4	0.729	0.636	91.67	0.3582531
		xalan-2.4	0.531	0.571	84.81	0.1618251
		xalan-2.5	0.601	0.569	60.2	0.2024827
		xalan-2.6	0.703	0.655	69.98	0.4046577
		xerces-1.3	0.508	0.25	85.46	0.0442431
		xerces-1.4	**0.884**	**0.935**	**92.81**	**0.8027267**
4	linear model	ant1.7	0.836	0.658	85.62	0.466764048
		camel1.4	0.632	0.444	85.96	0.1449815
		camel1.6	0.663	0.389	79.33	0.1070268
		camel1.2	0.634	0.618	62.3	0.1700227
		prop-4	0.503	0.138	89.56	0.020491
		xalan-2.4	0.503	0.138	83.97	0.020491
		xalan-2.5	0.53	0.541	53.73	0.0745921
		xalan-2.6	0.53	0.541	48.17	0.0745921
		xerces-1.3	0.56	0.125	79.12	-0.0158431
		xerces-1.4	**0.879**	**0.922**	**92.88**	**0.8146373**
5	neural network	ant1.7	0.75	0.221	79.26	0.030830877
		camel1.4	0.564	0.189	30.66	0.0507828
		camel1.6	0.564	0.189	29.72	0.0507828
		camel1.2	0.632	0.293	45.65	0.07284
		prop-4	**0.764**	**0.455**	**90.62**	**0.1217626**

		xalan-2.4	0.542	0	82.41	-0.0702759
		xalan-2.5	0.524	0.512	58.9	0.0183153
		xalan-2.6	0.584	0.547	43.1	0.1745973
		xerces-1.3	0.589	0.15	22.53	0.0422285
		xerces-1.4	0.587	0.777	45.54	0.0895409
6	Svm	ant1.7	0.746	0.214	78.55	0
		camel1.4	0.663	0.5	84.21	0.0636155
		camel1.6	0.748	0.667	82.19	0.1000923
		camel1.2	0.596	0.265	58.2	-0.0739102
		prop-4	0.596	0.265	79.7	-0.0739102
		xalan-2.4	**0.899**	**1.304**	**84.83**	**0.2937467**
		xalan-2.5	0.899	1.299	53.11	0.2937467
		xalan-2.6	0.507	0.466	52.39	0.0178312
		xerces-1.3	0.507	0.466	60.85	0.0178312
		xerces-1.4	0.512	0.749	60.59	0.0097906
7.	Random forest	ant1.7	0.66	0.624	76.91	0.72531487
		camel1.4	0.725	0.625	80.34	0.1887846
		camel1.6	0.620	0.417	81.94	0.168867
		camel1.2	0.735	0.462	77.34	0.1076602
		prop-4	0.803	0.58	89.23	0.2331917
		xalan-2.4	0.46	0.522	76.21	0.2714224
		xalan-2.5	0.739	0.621	55.89	0.2240989
		xalan-2.6	0.850	0.777	67.02	0.4667699

		xerces-1.3	0.662	0.5	82.07	0.4135098
		xerces-1.4	**0.942**	**0.933**	**94.60**	**0.7723019**
8.	MLP	ant1.7	0.565	0.437	72.82	0.6263481
		camel1.4	0.5274	0.357	83.07	0.0976947
		camel1.6	0.532	0.294	77.43	0.0996254
		camel1.2	0.519	0.397	60.98	0.0464367
		prop-4	**0.59**	**0.557**	**92.87**	**0.2684842**
		xalan-2.4	0.624	0.341	81.77	0.1971909
		xalan-2.5	0.628	0.58	62.19	0.2655879
		xalan-2.6	0.751	0.726	70.65	0.4138944
		xerces-1.3	0.592	0.5	87.67	0.2706026
		xerces-1.4	0.922	0.628	91.53	0.7830603
9.	Boosting	ant1.7	0.893	0.519	80.08	0.372531487
		camel1.4	0.628	0.625	83.95	0.1887846
		camel1.6	0.691	0.417	79.94	0.168867
		camel1.2	0.598	0.462	67.34	0.1076602
		prop-4	0.792	0.58	91.83	0.2331917
		xalan-2.4	0.657	0.522	88.19	0.2714224
		xalan-2.5	0.643	0.621	61.18	0.2240989
		xalan-2.6	0.808	0.777	73.24	0.4667699
		xerces-1.3	0.696	0.530	84.07	0.4135098
		xerces-1.4	**0.928**	**0.932**	**92.53**	**0.7723019**

Table 5.6 20-Features selected by Boruta

CHAPTER-6
CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The thesis work on Software Fault Prediction consists of 10 Datasets. The datasets contains the software modules and its features that stated which of the software modules were faulty and which were not. All the datasets used were classification datasets and were used with different Machine learning algorithms.

The algorithms used predicted the accuracy against the actual values of the datasets and furthermore the errors were calculated using different formulas. The error rate helped to find out which algorithms were much more useful than the others. The Algorithms used were Random forest, Linear regression, Neural network, Adaboost, Bagging using Linear regression, SVM .

Kruskal wallis test was applied using the AAE(Average absolute error) and ARE(Average relative errors). Kruskal test is commonly used in case of where the first variable is a nominal variable and the other one is measurement variable, also we can say that it works as one-way ANOVA does.

Also, for all the datasets used the parameters such as Sensitivity, Accuracy, MCC, ROC were also predicted using feature selection on each of the datasets. The feature selection was done using an algorithm called BORUTA. 10, 16, 20 features were taken in account for calculation the above mentioned parameters. The parameters depicts different results against different features. The algorithms which performed well on all the datasets on average were Bagging, SVM , Adaboost and neural network.

6.2 Future Scope

In the Proposed thesis, only the classification algorithms have been implemented. However, in future new and different machine learning methods and models can be tried in the environment. More new and important attributes could be added to the currently used datasets to expand the effectiveness of the models that we have used. We will try to use more different datasets and some other techniques and methods to simplify and generalize our finding. Moreover, in forthcoming time we will emphasise on using different domain datasets and then evaluating and validating some new models for the number of faulty and non-faulty modules prediction.

REFERENCES

[1] Rathore, S. Singh, and S. Kumar. "An empirical study of some software fault prediction techniques for the number of faults prediction." *Soft Computing* 21.24 (2017): 7417-7434.

[2] Tiwari, A. Kumar. "Machine learning based approaches for prediction of Parkinson disease." *Mach Learn Appl* 3.2 (2016): 33-39.

[3] P. Singh Comprehensive model for software fault prediction. InInventive Computing and Informatics (ICICI), International Conference on 2017 Nov 23 (pp. 1103-1108). IEEE.

[4] S. Dhankhar , H. Rastogi , M. Kakkar "Software fault prediction performance in software engineering." InComputing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on 2015 Mar 11 (pp. 228-232). IEEE.

[5] L. Chen , B. Fang, Z. Shang . "Software fault prediction based on one-class SVM". InMachine Learning and Cybernetics (ICMLC), 2016 International Conference on 2016 Jul 10 (Vol. 2, pp. 1003-1008). IEEE

[6] Y. Jiang , J. Lin , B. Cukic , S. Lin , Z. Hu "Replacing code metrics in software fault prediction with early life cycle metrics." *Information Science and Technology (ICIST), 2013 International Conference on*. IEEE, 2013.

[7] Bishnu, S. Partha , and V. Bhattacherjee. "Software fault prediction using quad tree-based k-means clustering algorithm." *IEEE Transactions on knowledge and data engineering* 24.6 (2012): 1146-1150.

[8] S. Karim , HL. Warnars , FL. Gaol , E. Abdurachman , B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset." *Cybernetics and Computational Intelligence (CyberneticsCom), 2017 IEEE International Conference on*. IEEE, 2017.

[9] Yang, Ning, X. Zhao, and H. Zhang. "2012 8th International Conference on Natural Computation (ICNC)."

[10] Kumudha, P., and R. Venkatesan. "Cost-sensitive radial basis function neural network classifier for software defect prediction." *The Scientific World Journal* 2016 (2016).

[11] G. Boetticher, T. Menzies, T. Ostrand,"The PROMISE repository of empirical software engineering data." *http://promisedata. org/repository* (2007).

[12] A. Shanthin , R.M. Chandrasekaran "Applying machine learning for fault prediction using software metrics." *International Journal of Advanced Research in Computer Science and Software Engineering* 2.6 (2012): 274-284.

[13] T. Hall, S. Beecham , D. Bowes, D. Gray , S. Counsell "A systematic literature review on fault prediction performance in software engineering." *IEEE Transactions on Software Engineering* 38.6 (2012): 1276-1304

[14] Gayathri, M., and A. Sudha. "Software defect prediction system using multilayer perceptron neural network with data mining." *International Journal of Recent Technology and Engineering* 3.2 (2014): 54-59.

[15] J. Li, P. He, J. Zhu, and M. R. Lyu, “Software Defect Prediction via Convolutional Neural Network,” *2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur.*, pp. 318–328, 2017.

[16] Rathore, S.Singh, and S. Kumar. "A decision tree regression based approach for the number of software faults prediction." *ACM SIGSOFT Software Engineering Notes* 41.1 (2016): 1-6.

[17] Gao, Kehan, and T. M. Khoshgoftaar. "A comprehensive empirical study of count models for software fault prediction." *IEEE Transactions on Reliability* 56.2 (2007): 223-236.

[18] Li, Wei, and S. Henry. "Object-oriented metrics that predict maintainability." *Journal of systems and software* 23.2 (1993): 111-122.

[19] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing,* vol. 27, pp. 504-518, 2015.

[20] T. J. McCabe, "A complexity measure," *Software Engineering, IEEE Transactions on,* pp. 308-320, 1976.

[21] Murphey,L. Yi , H. Guo, and L. A. Feldkamp. "Neural learning from unbalanced data." *Applied Intelligence* 21.2 (2004): 117-128.

[22] Fayyad, M. Usama , and K. B. Irani. "On the handling of continuous-valued attributes in decision tree generation." *Machine learning* 8.1 (1992): 87-102.

[23] G. Succi , W. Pedrycz,S. Djokic, P. Zuliani, B. Russo"An empirical exploration of the distributions of the Chidamber and Kemerer object-oriented metrics suite." *Empirical Software Engineering* 10.1 (2005): 81-104.

[24] V.U. Challagulla ,F.B. Bastani ,I.L. Yen , R.A. Paul "Empirical assessment of machine learning based software defect prediction techniques." *International Journal on Artificial Intelligence Tools* 17.02 (2008): 389-400.

[25] Grbac, T. Galinac, G. Mausa, and B. Dalbelo. "Stability of Software Defect Prediction in Relation to Levels of Data Imbalance." *SQAMIA*. 2013.

[26] S. Kim , H. Zhang ,R. Wu,L. Gong "Dealing with noise in defect prediction." *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 2011.

[27] M. Tan , L. Tan, S. Dara , C. Mayeu "Online defect prediction for imbalanced data." *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*. Vol. 2. IEEE, 2015.

[28] G. Calikli,A. Bener. "An algorithmic approach to missing data problem in modeling human aspects in software development." *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*. ACM, 2013.

[29] J.G. Moreno-Torres,T. Raeder, R . Alaiz-RodríGuez, N.V. Chawla, F. Herrera "A unifying view on dataset shift in classification." *Pattern Recognition* 45.1 (2012): 521-530.

[30] T.L. Graves, A.F. Karr, J.S. Marron, H. Siy,"Predicting fault incidence using software change history." *IEEE Transactions on software engineering* 26.7 (2000): 653-661.

[31] D. Rodriguez ,R. Ruiz, J. Cuadrado-Gallego, J. Aguilar-Ruiz ,M. Garre "Attribute selection in software engineering datasets for detecting fault modules." *Software*

Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on. IEEE, 2007

[32] K . Gao, T.M. Khoshgoftaar, H. Wang, N. Seliya "Choosing software metrics for defect prediction: an investigation on feature selection techniques." *Software: Practice and Experience* 41.5 (2011): 579-606.

[33] L. Xuan ,C. Zhigang ,Y. Fan "Exploring of clustering algorithm on class-imbalanced data." *Computer Science & Education (ICCSE), 2013 8th International Conference on*. IEEE, 2013.

LIST OF PUBLICATIONS