Evaluation of Feature Selection Techniques for Software Maintenance Prediction

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

Master of Engineering

in

Computer Science and Engineering

Submitted By

Sheena Nanda

(Roll No. 801532047)

Under the supervision of:

Dr. Anju Bala

Assistant Professor

Dr. Sharad Saxena

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2017

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, ***"Evaluation of Feature Selection Techniques for Software Maintenance Prediction"***, 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 University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Anju Bala & Dr. Sharad Saxena*** 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.

Sheena Nanda

(Sheena Nanda)

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

Anju Bala

(Dr. Anju Bala)
Assistant Professor, CSED

(Dr. Sharad Saxena)
Assistant Professor, CSED

ACKNOWLEDGEMENT

I acknowledge my debt to those who have contributed significantly to my efforts in this research work and dissertation. I would like to express deep sense of gratitude to **Dr. Anju Bala**, Assistant Professor, CSED & **Dr. Sharad Saxena**, Assistant Professor, CSED who have been a great source of inspiration, guidance and moral support for me. It would never be possible for me to continue this study without their constant support, encouragement and positive attitude. It has been a great pleasure and experience working with them.

I am also grateful to **Dr. Maninder Singh**, Associate Professor and Head, CSED for his inspiration. He sets high principles for his students and motivates and guides them to meet those principles.

Before ending I would like to thank my parents and friends for their love, motivation, support and blessings. They have been a constant source of love, concern, support and strength for me all these years.

Finally I would like to thank the management of Thapar University for providing me a great opportunity for learning, not just in academics but also in many other creative things.

I sincerely regret any inadvertent omissions. With my heartiest thanks to all.

Date: 18th July, 2017 **(Sheena Nanda)**

Place: Thapar Univeristy, Patiala **Roll No. 801532047**

ABSTRACT

Software quality is the ease with which a process or software fulfills customer's expectation. One of the essential steps in measuring the quality of the software is software maintenance. It has been found that software maintenance accounts for 60-70% of the total cost thus, it is very important to prepare an accurate software maintenance plan during the software development. This helps to analyze the cost and risk associated with the software well in advance so that an optimized resource planning can be done. The development of a software is associated with high variance on techniques and goals; therefore, it is hard to measure software's quality. Hence, object-oriented metrics have turned into an essential part of software development process for quantitative measurement. However, not all the metrics affects the quality in same way. Feature selection methods decreases computation time by better understanding the data using machine learning. Hence, the objective of this dissertation has been to construct improved prediction model by pre-processing the data using feature selection techniques. Feature selection algorithms decrease the input variables by selecting the most relevant ones and removing the redundant variables. In this dissertation, `CHANGE' is used as the measure of maintainability. Two open source softwares: `Apache Log4j' and `Drumkit' have been considered for this research work and input variables have been extracted using CKJM and LocMetric tools. Further, JarComp tool has been used to calculate CHANGE per class among two versions of the open source software systems. Thereafter, most relevant features were extracted using seven different feature selection techniques and further, their effectiveness was measured using nine machine learning algorithms for software maintenance prediction. Subsequently, different combinations of feature selection techniques and prediction models have been analyzed to identify the best combination.

This dissertation will enable software developers to predict the change requirements in advance using software metrics that are relevant and affect the efficiency of the software which further helps to improve the quality of software thus, encouraging good coding and designing techniques.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SDLC	Software Development Lifecycle
ML	Machine Learning
FS	Feature Selection
MLR	Multiple-Linear Regression
GA	Genetic Algorithm
GMDH	Group Method of Data Handling
UIMS	User Interface Management System
QUES	Quality Evaluation System
MARS	Multivariate Apadtive Regression Splines
ANN	Aritifical Neural Network
FSS-EBNA	Feature Subset Selection-Estimation of Bayesian Network Algorithm
EDA	Estimation of Ditribution Algorithm
SFFS	Sequential Forward Floating Selection
r	Coefficient of Correlation
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
C&K	Chidamber & Kemerer
MOOD	Metrics for Object-Orinted Design
QMOOD	Quality Model for Object-Oriented Design
FoBa	Forward Backward

CHAPTER – 1 INTRODUCTION

In this ever changing world, the requirements of the customer keep changing, thus, the software also needs to be changed. The change in software can be due to change in technology, introduction of new hardware or enhancement of the functionality. The ease with which software can adapt to the changing requirements forms an essential component of software maintenance. It is an expensive task as compared to software development in terms of resources, effort and time as it lasts for about 5-6 years after the software development. Therefore, it is very essential to produce software that is easy to maintain in future.

1.1 Software Maintenance

Modification of software to correct bugs, enhance performance or other parameters after delivery to the customer is termed as software maintenance. The total cost and risk associated with the software can be estimated using software maintenance. It can incur almost 70% of the total effort and cost expended on the software in its lifetime [1]. Most of the people have an impression that software maintenance is only about fixing the bugs. But, there are many reports submitted by users that suggest there are more functional enhancements as compared to the bugs. The different reasons for modifications in the software are:

- **Changed Market Conditions:** There may be change in market conditions, like taxation policies, maintain bookkeeping etc.
- **Change in Customer's Requirements:** Customer may change his requirements or like to add new features to the software.
- **Change in Technology:** If there is any change in technology, platform or hardware software needs to adapt to it.
- **Change in Organization Hierarchy:** There may be change in organization structure of the customer like reduction in human resources; acquisition of new company, the original software may need to change.

Software maintenance, which can keep going for a long time (or even decades) after the advancement procedure, requires a viable arrangement which can address the extent of programming upkeep, the fitting of the post conveyance/organization handle, the assignment of who will give support and a gauge of the life-cycle costs. The determination of appropriate implementation of guidelines is a testing errand ideal from early phase of software designing which lacks unmistakable significance by the concerned partners.

The different categories of software maintenance in ISO/IEC 14764 are:

- **Corrective Maintenance:** This incorporates changes and updating done with a specific end goal to right or fix issues, which are either found by client or concluded by client mistake reports.
- **Adaptive Maintenance:** This includes modifications and updations applied to stay the software package up-to date and tuned to the ever ever-changing world of technology and business surroundings.
- **Perfective Maintenance:** This incorporates adjustments and updates done with a specific end goal to keep the product usable over drawn out stretch of time. It incorporates new elements, new client necessities for refining the product and enhance its unwavering quality and execution.
- **Preventive Maintenance:** This incorporates alterations and updations to avert future issues of the product. It plans to go to issues, which are not critical right now but rather may cause difficult issues in future.

Software maintainability is a vital code quality attribute. It quantifies the level of simplicity with which the product is updated, repaired or perceived. It can only be measured once the software is operational for certain time duration. Anticipating software maintainability ahead of time is useful in breaking down the cost and hazard related with the product and improves asset arranging. Hence, the maintainability of a product can altogether affect software costs. This implies it is critical to have the capacity to gauge a product’s maintainability in order to adequately oversee costs.

Maintenance and maintainability are two distinct concepts yet closely related to each other. Maintenance is an essential step in Software Development Lifecycle (SDLC) process whereas maintainability finds the quality of the software. It implies that maintenance corresponds to process while maintainability on the other hand corresponds to quality. The prediction of cost incurred to maintain the software is called maintenance cost prediction and estimations made to measure the quality is called quality characteristic measurement. Section 1.2 discussed software maintenance prediction in detail.

1.2 Software Maintenance Prediction

Software industry acknowledges software maintenance prediction as a complex process. It has been found that software maintenance accounts for 60-70% of the total cost thus, it is very important to prepare an accurate software maintenance plan during the software development. This helps to analyze the cost and risk associated with the software well in advance so that an optimized resource planning can be done.

Prediction or estimation plays an important role in project planning. Estimations can not only be done for processes or products but also projects. The procedure to measure effort involved in the project is referred to as effort estimation. Further, the process to assess the maintenance procedure is called as project maintenance effort estimation or maintenance cost estimation. Consequently, appraisals of quality characteristics give a quantifiable estimation of the quality of the features that a product item has.

A software maintenance forecasting algorithm empowers institutions to foresee the maintainability of their product, thus helping to deal with their assets and resources in an efficient manner. This additionally helps to lessen the effort required for maintenance and along these lines diminishing the general expenditure and time spent on a software project.

1.3 Machine Learning Algorithm

Machine learning (ML) constructs a model using a particular algorithm where in some percentage of data is used as training data and remaining as testing data. ML enables the system to get into a self-learning model without programming it explicitly. When the algorithm is exposed to new data, it learns, grows, changes and develops by itself. The basic steps involved in the process of ML are depicted in Fig. 1.1 below. The historical data available is pre-processed and divided into training and testing data. Further, the ML algorithm, dataset and validation techniques work together to produce the final prediction model.

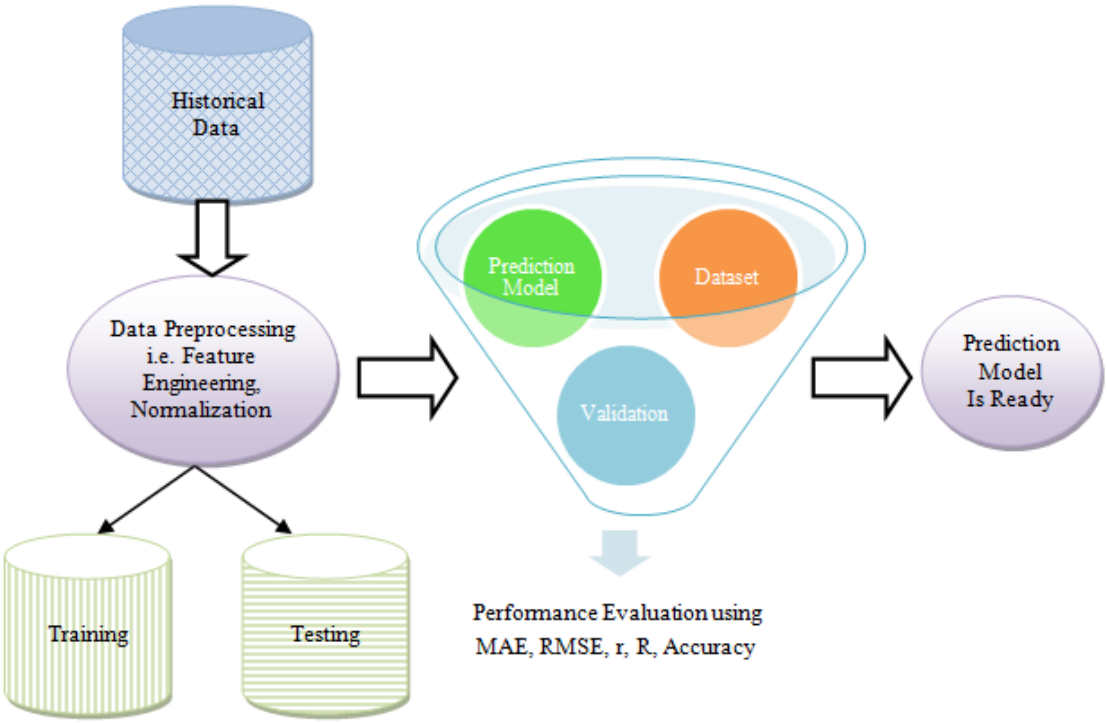

Fig. 1.1 Machine Learning Process

The different types of learning (supervised and unsupervised) are elucidated below:

- **Supervised Learning:** In this learning the training set consists of pair of input variables and the desired output value. It consists of an inferred function that uses labeled data set for mapping. New dataset cab be mapped from this inferred function.

- **Unsupervised Learning:** The training set consists of only input variables and no desired output value in case of unsupervised learning. It uses unlabeled data sets to describe the hidden patterns.

ML systems are being connected to an extensive variety of areas for research and business examination and include recognizing patterns and rolling out improvements in the algorithms. In this day and age, the information utilized for ML algorithms has expanded exponentially, thus, applying feature selection (FS) systems for picking a subset of critical measurements is vital. Therefore, the various FS types have been discussed in the next section 1.4 in order to reduce the data dimensionality.

1.4 Feature Selection Algorithm

A feature is an individual quantifiable property of the process being watched. Any ML algorithm can perform categorization utilizing a set of features. The space of features has extended from tens to several features or elements utilized as a part of the applications of ML or pattern recognition. A few systems have been produced to address the issue of lessening unimportant and repetitive features which are a burden on challenging tasks.

FS is the process of filtering relevant features and leaving behind redundant features, thus, building better prediction model with reduced cost and better understanding of the underlying processes that generated the data. It reduces the impacts of noise and unessential features without hampering the estimation accuracy and affix the estimation procedure. The basic algorithm of FS is to find the relevance of each feature with the output variable and then eliminate the features with low relevance [2]. FS does not create new features rather it uses the input features itself to decrease their number. Once a FS is chosen, a method is created which finds the subset of valuable features. Specifically assessing every one of the subsets of features for given information turns into a NP-hard issue as the number of features increases [2]. Subsequently, suboptimal procedure is utilized which can expel repetitive data with tractable calculations. FS techniques are comprehensively characterized into filter and wrapper strategies [3]. The FS algorithms are broadly classified as: Filter Methods,

Embedded Methods and Wrapper Methods as depicted in the hierarchy in Fig. 1.2. The details of different types of FS algorithms have been discussed in further sub-sections.

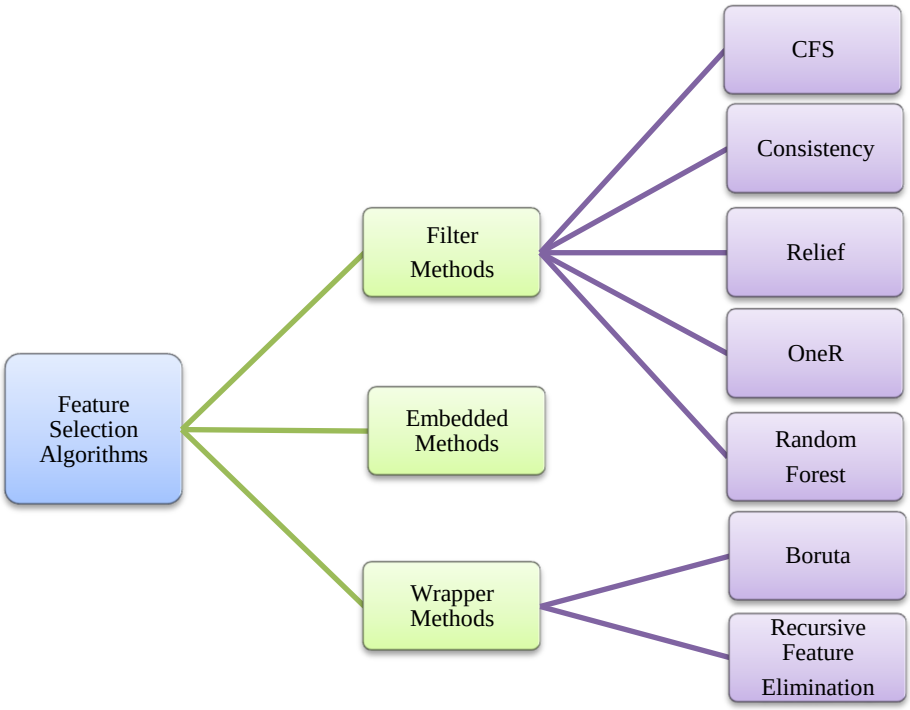

Fig. 1.2 Hierarchy of Feature Selection Algorithms

1.4.1 Filter Methods

Filter methods are utilized for pre-handling to rank the features wherein the much positioned features are chosen and connected to an output. They utilize variable positioning methods as the principle indicator for variable determination by sorting using ranking methods which are used because of their effortlessness and good achievement rate. An appropriate ranking method like correlation, mutual information and so on is utilized to assign rates to the features and a threshold is set to remove features beneath it. Ranking methods lie in the category of filter methods since they are activated before classification to filter irrelevant variables.

1.4.2 Wrapper Methods

Wrapper methods wrap around the search algorithm and find the subset of features that gives the best performance. The performance is measured by the ability to have high prediction accuracy rate. Input variables are used as black box and the objective function to assess the prediction accuracy is used as predictor's performance. It has to evaluate 2^N which turns out to be NP-Hard problem. Hence, the search algorithm uses heuristic approach to find the feature subset resulting in suboptimal set. There are numerous search algorithm proposed that finds the subset keeping in mind the optimization of objective function. Branch and bound works on the principle of tree structure. But this approach does not work well when number of features increases as it results in exponential increase in search. Therefore, simplified algorithms like sequential search and evolutionary algorithms like GA etc. are used which yield better results.

Wrapper methods can be broadly classified in the following two categories:

- **Sequential Selection Algorithms:** It operates on an initial empty set by sequentially adding features with the intent to maximise the objective function. It can also initially include all the features in a set and then sequentially decrease the features that hinder the maximisation of the objective function. To accelerate the final set creation, a criterion is picked which incrementally expands the objective function until the most desirable objective function is achieved with least number of features.

- **Heuristic Search Algorithms:** In this method a function called, heuristic function is defined, that ranks the alternate subsets based on the information available at every branching step so as to choose the branch to follow. Heuristic method may not lead to the best solution but is still a valuable as it is not too much time consuming.

1.4.3 Embedded Methods

Embedded techniques incorporate variable determination as a component of the preparation procedure without parting data into training and testing sets [4]. Their aim is to diminish the calculation time taken up for reclassifying different subsets as done in wrapper strategies [5]. The principle approach is to fuse the feature determination as a part of the preparation procedure (training process).

CHAPTER – 2 LITERATURE REVIEW

In this chapter the work done by various researchers in the field of software maintenance prediction has been discussed. Software engineering involves producing high quality software with predictable cost and schedule. The main aim of software engineering is to control the cost, schedule and quality of the software. Hence, the concept of software metrics came into existence [6]. Section 2.1 below discusses the work done by different researchers to prove that object-oriented metrics have strong relation with software maintenance prediction.

2.1 Object-Oriented Metrics & Software Maintenance Prediction

Object-oriented metrics play an important role to measure the quality and development process of the software quantitatively. These metrics covers the key concepts of object-oriented programming: methods, classes, coupling, cohesion and inheritance**.** It has been empirically proven that there is a strong connection between the object-oriented metrics and software maintenance prediction.

Li [7] proposed a new suite of object-oriented metrics after analyzing the object-oriented metrics proposed by Chidamber and Kemerer [8] i.e. WMC, DIT, NOC, CBO, RFC and LCOM. Various researchers have used these metric to predict maintainability of object-oriented software system by measurement of maintenance effort.

Aggarwal et al [9] explored twenty-two object-oriented metrics proposed by different analysts. They characterized the metrics and furthermore clarified those utilizing handy applications. Moreover, they connected the metrics on standard projects on the premise of which descriptive statistics, principal component analysis and correlation analysis was exhibited. At long last, an audit of the exact review concerning picked metrics and subset of the measures that give adequate data was given and metrics giving overlapping information were avoided from the set.

Riaz et al [10] played out a deliberate audit of software maintainability prediction and metrics in light of fifteen reviews. The review was focused at the software quality characteristic of maintainability rather than the procedure of software maintenance.

Fioravanti and Nesi [11] conferred a model and metrics for estimation/prediction of adaptive maintenance effort. The model projected may be used as a general procedure for short listing distinguished metrics for forecasting adaptive maintenance effort. The model and metrics projected are valid against real knowledge by victimization multi-linear multivariate analysis.

2.2 Software Maintenance Prediction using Machine Learning

ML is utilized for information investigation which helps in mechanizing development of a prediction model. An extensive variety of experimental reviews have been done that recommend the utility and utilization of ML calculations in numerous areas. Different prediction models and object-oriented metrics have been proposed for software maintenance prediction.

Li and Henry [6] made a conclusion that object-oriented metrics have a solid impact on maintainability effort and a combination of metrics collected from the software's source code can be used as input for predicting maintenance effort which they have done using Multiple Linear Regression (MLR) Model.

Malhotra and Chug [12] additionally built proper model utilizing ML algorithms for foreseeing maintainability of object-oriented software that can be connected effectively and additionally gauge with least mistakes. They fabricate their prediction model utilizing distinctive ML algorithms, for example, Genetic Algorithms (GA) and Group Method of Data Handling (GMDH). They assessed the execution of this model utilizing User Interface Management System (UIMS) and Quality Evaluation System (QUES) dataset and reasoned that GMDH network model is one of the best demonstrating methods to anticipate the product maintainability.

Zhou and Leung [13] forecasted object-oriented software maintainability utilizing Multivariate Adaptive Regression Splines (MARS) and contrasted their outcomes and other prediction models. They used Li and Henry's datasets i.e. UIMS and QUES for results comparison. In their maintainability prediction model they utilized CHANGE metric and LOC number changed amid the maintenance time frame.

Heiat [14] played out a correlation between artificial neural system (ANN) and regression models for evaluating software development effort. He thought about the prediction efficiency of Multilayer Perceptron and Radial Basis Function neural systems with other regression models, demonstrating that when a consolidated third era and fourth era languages data sets is utilized, the neural system created indicated enhanced execution over conventional regression analysis considering mean absolute error.

Kaur and Kaur [15] did a factual examination of modeling methods for software maintainability prediction by building models utilizing twenty-seven distinctive regression and ML based algorithms. The results were compared and analyzed using two object-oriented systems i.e. UIMS and QUES developed using an object-oriented programming language Ada.

Dubey et al [16] used Li-Henry object-oriented metrics i.e. DIT, LCOM, DAC and NOM to predict the maintainability using Multi-Layer Perceptron neural network model. They used neural network with three input layers, two hidden layers and one output layer. Their prediction model was built using UIMS dataset which showed accurate results.

Ahmed and Hamdi [17] proposed a procedure for creating fuzzy-based straightforward model for estimating software maintainability. They connected the procedure to a contextual analysis where Mamdani fuzzy inference engine is utilized to estimate software maintainability and contrasted it and other ML T-S-based, SVM, PNN, RBF, BN and MARS models.

Aggarwal et al [18] proposed an ANN based model and anticipated the nature of software by figuring the CHANGE metric that is characterized as number of lines

changes per class. In this model, Li-Henry metrics were utilized to foresee the maintainability of software. The model was prepared utilizing back-propagation algorithm and testing and examination of the outcomes accumulated demonstrated that this model is helpful in predicting the maintenance exertion of the software.

Koten and Gray [19] utilized a Bayesian network based viability forecast model which can be utilized for an object-oriented software system. This model is built utilizing Li-Henry model metrics which were gathered from various object-oriented systems. In this model they developed an extraordinary sort of Bayesian network called Naïve Bayes classifier which is a single node speaking to a classification variable and is associated with every single other node that speak to prediction factors. In this model, they utilized UIMS and QUES datasets to assess the proposed model exactness and the outcomes demonstrated that this Bayesian network based model (Koten-Gray model) have better precision as compared to regression analysis based models.

2.3 Machine Learning with Feature Selection

FS strategies can be joined with learning algorithms to beat the issue of managing tremendous set of features, some of which are not in any case important for analyzing efficiency in adequate time length.

Chandrasekhar and Sahin [2] presented and investigated stability of different FS procedures. They introduced to variable elimination which can be connected to a wide exhibit of ML issues and concentrated on Filter, Wrapper and Embedded FS techniques. Further, they connected the FS methods on standard datasets to show their appropriateness.

Kohavi and John [3, 20] talked about wrapper techniques for FS. They concentrated on the qualities and shortcomings of the wrapper approach and demonstrated a progression of enhanced plans. Furthermore, they contrasted the wrapper approach with induction without FS and to Relief, a channel way to deal with FS. Huge change in precision was accomplished by them for some datasets for the two groups of

induction algorithms utilized: decision trees and Naive - Bayes. They additionally depicted a strategy for FS utilizing cross-validation that is appropriate on any induction algorithm and talked about the after effects of ID3 and C4.5 on artificial and genuine datasets.

Hall and Holmes [21] played out an examination of a few feature selection techniques for supervised learning wherein the strategies delivered a feature positioning/ranking which is useful in isolating the value of a feature separately. Feature subset selection is accomplished by cross-validating the feature rankings concerning a classification learner to locate the best features. They announced outcomes for a selection of standard data sets and two diverse learning algorithms C4.5 and Naïve-Bayes.

Chen et al [22] did a broad review on wrapper techniques for FS and reasoned that evacuating the unimportant elements help in expanding the prediction model's proficiency. Their general objective was to empower repeatable, refutable and improvable analyses in software programming, in this way, they utilized open source cost model (COCOMO) and freely accessible datasets.

Manchanda and Chug [23] dissected different combinations of feature subset selection and ML algorithms. They recognized metrics that constitute the littlest conceivable set to decide and control the requirement for new versions to enhance the basic nature of the software product. It was watched that the miscalculation in change estimates is lessened in the wake of applying different FS strategies and the effort required to break down littler set of metrics is lesser than that required to analyze all the metrics to foresee the CHANGE for an extensive scale project.

Dash and Liu [24] have performed FS wherein their work concentrated on irregularity measure as indicated by which a feature subset is conflicting if no less than two cases with same feature value however with various class labels are available. They compared irregularity measure with different measures and concentrated distinctive search techniques like exhaustive, complete, heuristic and random search; that can be connected to this measure. Further, they played out an exact review to look at the focal points and weaknesses of these search techniques, give rules on picking a

specific search strategy and analyze the classifier error rates previously, then after the fact FS.

Inza et al [25] proposed a FS technique by Estimation of Bayesian Network Algorithm (FSS-EBNA) in which they selected wrapper method above Naive-Bayes and ID3 calculations for estimation. FSS-EBNA uses randomized search technique, is populace based and can be applied in the absence of domain information. FSS-EBNA depends on the EDA (Estimation of Distribution Algorithm) paradigm and along these lines, maintains a strategic distance from the utilization of crossover and mutation operators to develop the populace rather than Genetic Algorithms.

Jain and Zongker [26] demonstrated that Sequential Forward Floating Selection (SFFS) algorithm is one of the predominant FS techniques by assessing SFFS on land use classification. They concentrated the issue of picking an ideal list of features for land utilize order in light of SAR satellite pictures utilizing four distinctive texture models. Pooling features obtained from various texture models, trailed by a FS brought about a considerable change in the classification precision.

Chug and Malhotra [27] used object-oriented and genetic FS on open source software to lead an extensive review on prediction of maintainability. They directed substantial scale experimental examinations on seven open source software dataset based on thirteen classifiers. Thereafter, statistical tests and post hoc investigation was done to affirm the execution of one ML strategy over another. They analyzed the execution of ML systems utilizing four prediction accuracy measures Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Pred(0.25) and Pred(0.75). The diversity among the performances of different ML systems was additionally evaluated using Friedman test and Nemenyi test to recognize whether there exists measurable difference of performance between sets of various ML strategies.

Gunnalan et al [28] exhibited another technique TAR2 treatment learner for FS. TAR2 expect little spines; i.e. few components will be sufficient for choosing favored classes. TAR2 can be utilized as a pre-processor to different learners for recognizing valuable feature subsets.

Khalid et al [29] led a review on FS strategies in ML with the goal of analyzing how viably these systems can be utilized to accomplish superior of learning algorithms that at last enhances performance efficiency of a classifier. The review was led on seven datasets including lung malignancy, leukemia and five other datasets from UCI ML.

The analysis and research done by different authors in software maintenance prediction domain discussed in the above sub-sections have been summarized in TABLE 2.1

TABLE 2.1 Methods/Model Classification

S.No.	Author	Year	Method/Model Used	Reference
1.	W. Li	1998	Metric Suite for Object-Oriented Programming	[7]
2.	S. Chidamber and C.F. Kemerer	1994	Metric Suite for Object-Oriented Design	[8]
3.	K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra	2006	Empirical Study of Object-Oriented Metrics	[9]
4.	M. Riaz, E. Mendes and E. Tempero	2009	Systematic Review of SMP and Metrics	[10]
5.	F. Fioravanti and P. Nesi	2001	Multi-Linear Regression Analysis	[11]
6.	W. Li and S. Henry	1993	Object-Oriented Metrics that Predict Maintainability	[6]
7.	R. Malhotra and A. Chug	2012	Genetic Algorithms (GA) and Group Method of Data Handling (GMDH)	[12]
8.	Y. Zhou and H. Leung	2007	Multivariate Adaptive Regression Splines (MARS)	[13]
9.	H. Heiat	2002	Artificial Neural Network (ANN)	[14]
10.	A. Kaur and K. Kaur	2013	Comparison of Modeling Methods for SMP	[15]
11.	S.K. Dubey, A. Rana and Y. Dash	2012	Multilayer Perceptron Model	[16]
12.	M.A. Ahmed and A.A.J. Hamdi	2013	Fuzzy-Based Transparent Model	[17]
13.	K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra	2006	Artificial Neural Network	[18]
14.	C.V. Koten and A.R. Gray	2006	Bayesian Network	[19]

15.	G. Chandrashekar and F. Sahin	2014	Survey on Feature Selection Methods	[2]
16.	R. Kohavi and G.H. John	1997	Wrapper Methods for FSS	[3]
17.	G.H. John, R. Kohavi and K. Pfelgar	1994	Irrelevant Features and the Subset Selection Problem	[20]
18.	M.A. Hall and G. Holmes	2003	Attribute Selection Techniques	[21]
19.	Z. Chen, T. Menzies, D. Port and D. Boehm	2005	Right Data for Software Cost Modelling	[22]
20.	S. Manchanda and A. Chug	2015	CFS FSS Technique	[23]
21.	M. Dash and H. Liu	2003	Consistency Based FSS	[24]
22.	I. Inza, P. Larranaga, R. Etxeberria and B. Sierra	2000	FSS by Bayesian Network-Based Optimization	[25]
23.	A. Jain and D. Zongker	1997	Sequential Forward Floating Selection (SFFS) FSS Algorithm	[26]
24.	A. Chug and R. Malhotra	2016	FSS using Genetic Algorithm	[27]
25.	R. Gunnalan, T. Menzies, K. Appukutty and A. Srinivasan	2003	TAR2less	[28]
26.	S. Khalid, T. Khalil and S. Nasreen	2014	Relief, Correlation Based FSS Method, PCA and ICA	[29]

CHAPTER – 3 PROBLEM STATEMENT

3.1 Problem Statement

Software maintenance is a critical activity that begins once the project or software is delivered to the customer. The job of the development team does not end by deploying a working project at customer's site, but they also have to maintain the software for some agreed upon period, which may be in years. It is a false notion that maintenance is merely solving of bugs, maintenance also includes enhancements based on the customer's requirements. Now days, competition in the market is too high leading to maintenance becoming an integral part of software industry. Software quality is also measured using software maintenance. Software quality measures the extent to which the software meets customer's expectations including the changing requirements. Predicting the efforts, cost and time involved in the maintenance process is a challenging task as the parameters that affect software's quality could be many, thus, the first and foremost question has been tried to answer in this study is identifying the features/software metrics which affect maintainability prediction using two open source software systems. A large number of metrics are available that possibly influence maintainability but, not all actually have an impact on the process of maintainability prediction which is study is done using ML algorithms. The main focus has been to perform this research work on open source software systems so that more generalized results can be obtained.

3.2 Research Gaps

A lot of research has been done in the field of predicting software maintenance using ML but some research gaps were noticed during literature review which are discussed as follow:

- As it has been observed much work has not been done on software maintenance prediction with pre-processing of data using FS techniques. All

input variables/features do not affect the output variable [6, 12, 13, 14, 15, 16, 17, 18 and 19].

- In today's high dimensionality domains, the count of input parameters sometimes increases exponentially and thus, the prediction process overruns the processing budget, effort and duration [6, 12, 13, 14, 15, 16, 17, 18 and 19].
- As observed in the literature survey much work has not been done on open source software so far. Most of the researchers have considered Li and Henry's UIMS and QUES datasets for their analysis [6, 12 and 15].
- Further, the prior work done on Apache Log4j and Drumkit dataset does not include FS in them [27].
- It has not been established which particular ML algorithm works best with which FS technique [27, 23].

3.3 Research Objectives

In this study, the research gaps observed during literature review and discussed in the above section have been accomplished as follow:

- The subset of most significant and relevant features has been obtained which provide better and more accurate results.
- Using all the features for constructing the prediction model is expensive and wastage of time. Thus, feature reduction has been achieved by implementing various FS methods.
- Open source software systems have been used so obtain more generalized results which have better industrial validity.
- For predicting maintenance of Apache Log4j and Drumkit software in terms of Change FS algorithms have been applied for eliminating redundant and irrelevant features.
- Different ML algorithms give varying results for prediction accuracy measures with different FS techniques. Further, a perfect match of the underlying criteria's of FS and ML algorithms has been established.

CHAPTER – 4 RESEARCH METHODLOGY

In this chapter, the extensive empirical study involving two datasets of open source software namely Apache Log4j and Drumkit, seven FS algorithms – CFS, Consistency, Boruta, Relief, OneR, Random Forest and Recursive Feature Elimination and nine ML algorithms –Linear Regression, General Regression Neural Network, Decision Tree, Cubist, Ridge Regression with Variable Selection, LASSO, Elastic Net, Random Forest and Principal Component Analysis has been discussed. Open source software have been selected for this research work as they help in improving industrial relevance and validity of the study. The steps adopted in this research work are depicted in Fig. 4.1 and discussed below:

- Two consecutive versions 1.1.2 and 1.1.3 of open source software Apache Log4j and versions 1-0.5.0 and 1-0.6.0 of Drumkit are taken into consideration.
- Different software object-oriented metrics portraying attributes of the open source software were ascertained utilizing CKJM [30] tool and LocMetrics [31] tool.
- CHANGE (insertion (considered one), deletion (considered one) or adjustment/modification (considered two – one for expansion, one for cancellation) per class was assessed utilizing JarComp [32] Tool.
- From that point, the normalized CHANGE (NCHANGE) estimations of the two versions of Apache Apache Log4j and Drumkit were figured.
- After collecting enough features, most relevant and important features are extracted using seven FS techniques namely CFS, Consistency, Boruta, Relief, OneR, Random Forest and Recursive Feature Elimination.
- The different sets of features extracted by different FS methods are used as inputs to nine ML algorithms – Linear Regression, General Regression Neural Network, Decision Tree, Cubist, Ridge Regression with Variable Selection, LASSO, Elastic Net, Random Forest and Principal Component Analysis for predicting software maintenance.

- Different evaluation parameters i.e coefficient of correlation (r), MAE, RMSE and Accuracy are used for calculating errors and accuracy of the prediction model.
- The blends of FS strategies and ML algorithms were examined to find which FS strategy gives best outcomes with which ML method.

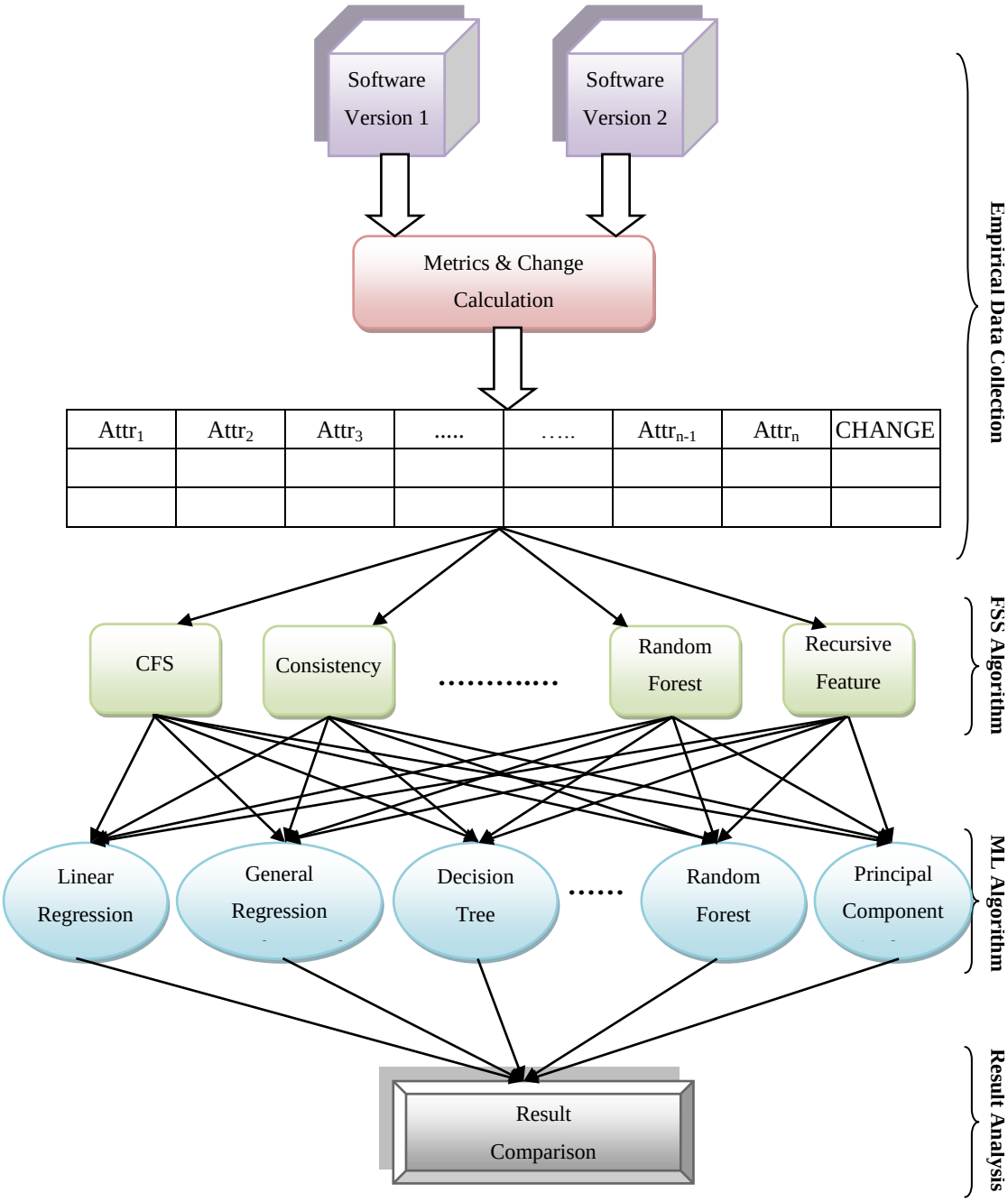

Fig. 4.1 Research Methodology

4.1 Object-Oriented Metrics

Quantitative measure to clarify at what degree a trait of testing or item quality or process has performed is known as software metrics. A considerable number of measures are utilized as a part of request to give the measurable information during the time spent in software development. Customary programming metrics goes for the procedure-oriented development due to which it can't satisfy the necessity of object-oriented software bringing about popularity of object-oriented design metrics in industrial software development environment as it aides in the improvement of higher quality items with minimal effort over their maintenance.

Object-oriented metrics define characteristics of object-oriented programming. Object-oriented metrics is fit for giving all the parameters to assess the complexity and quality related issues at the early development phase of a product.

Software metrics represent the structural and functional parts of a procedure, application or programming. Different reviews have presumed that object-oriented and maintainability have a solid affiliation [30, 31]. Software metrics potentially affect programming maintainability, especially the ones which survey the inter-connectivity of system components. In this way, they can foresee a scope of software features like CHANGE prerequisite.

Chidamber & Kemerer (C&K) gave six metrics that are commonly used. The metrics are: WMC, DIT, NOC, CBO, RFC and LCOM, these are calculated for each class. Various other object-oriented metric suits were also proposed by other researchers like MOOD (Metrics for Object-Oriented Design), QMOOD (Quality Model for Object-Oriented Design), Halstead's Complexity Measures, Henderson-Sellers Version etc.

4.1.1 CKJM Tool Metrics

There were six metrics that were initially proposed by Chidamber & Kemerer, but extended CKJM tool gives nineteen object-oriented metrics. It processes the byte code

of the compiled java file to find the metrics. The program evaluates for each class the metrics mentioned below and showcases them on its standard output or records them in a file.

- **WMC (C&K Metric Suite):** Weighted methods per class basically indicate the summation of McCabe's complexities of all local methods [9].

  WMC = summation of McCabe's cyclomatic complexity of all local methods; ranging from 0 to N; where N is a positive integer. (1)

- **DIT (C&K Metric Suite):** Depth of the inheritance tree indicates the level of a class in the hierarchy of inheritance. The effort in maintaining a class increases with the value of DIT metric [9].

  DIT = inheritance level number; ranging from 0 to N; where N is a positive integer. (2)

- **NOC (C&K Metric Suite):** Number of children metric measures the number of direct children a class has [8].

  NOC = number of direct sub-classes; ranging from 0 to N; where N is a positive integer. (3)

- **CBO (C&K Metric Suite):** Coupling between object classes gives a count of the efferent and afferent couplings to a specific class. A high CBO values is undesirable for efficient class maintainability [9].

- **RFC (C&K Metric Suite):** It stands for response of a class. It calculates the cardinality of the response set of a class wherein the response set includes all the local methods along with all the methods called by the local methods [9].

  RFC = number of local methods + number of methods called by local methods; ranging from 0 to N; where N is a positive integer. (4)

- **LCOM (C&K Metric Suite):** Lack of cohesion of methods metric measures the lack of cohesion of a class [8]. Cohesion indicates how tightly the local methods are linked to the local instance variables within a class.

  LCOM = number of disjoint sets of local methods; no two sets intersect; any two methods in the same set share at least one local instance variable; ranging from 0 to N; where N is a positive integer. (5)

- **Ca (Not a C&K Metric):** Afferent couplings indicate the number of other classes that make use of a given class.
- **Ce (Not a C&K Metric):** Efferent couplings indicate the number of other classes utilized by a given class.
- **LCOM3 (Henderson-Sellers Version Metric Suite):** This lack of cohesion in methods metric has values in the range of 0 to 2 and is calculated as:

$$LCOM3 = \frac{\left(\frac{1}{v}\ \sum_{k=1}^{v}\ \mu(M_k)\right) - p}{1 - p} \tag{6}$$

  Where, p – number of methods or procedures in class

  v - number of variables in class

  μ(M) - number of methods accessing a variable

- **NPM (Not a C&K Metric):** Number of public methods counts all the methods that are declared as public in a class.
- **LCO (Not a C&K Metric):** This lines of code metric gives a summation of number of fields, number of instructions in every method and number of methods of each class from the binary java code.
- **DAM (QMOOD Metric Suite):** It stands for data access metric and provides a ratio of number of private variables to the total number of variables in a specific class. Its value ranges between 0 and 1 and a high DAM value is desirable [33].
- **MOA (QMOOD Metric Suite):** Measure of aggregation metric evaluates the degree of part-whole relationships, realized by the using variables [33].
- **MFA (QMOOD Metric Suite):** The value of measure of functional abstraction metric lies between 0 and 1 and gives a ratio of the number of methods inherited by a class to the total number of methods available for access to the member methods of the class [33].
- **CAM (QMOOD Metric Suite):** Cohesion among methods of class ranges from 0 to 1 and is calculated by using the sum of different type of method parameters in all methods divided by the multiplication of number of different method parameter types in whole class and number of methods. It thus, finds the relatedness between methods of a class on the basis of parameter list of methods [33].

- **IC (C&K Metric Suite):** Inheritance coupling is a quality oriented expansion of the C&ampK metric suite. It gives a count of the parent classes to which a specific class is coupled.
- **CBM (C&K Metric Suite):** Coupling between methods evaluates the count of new/polished methods to which all the inherited methods are coupled.
- **AMC (C&K Metric Suite):** It stands for average method complexity and gives the average method size for each class.

4.1.2 LocMetrics Tool Metrics

LocMetrics tool calculates the following software metrics for each class: LOC: Total Lines of Code, BLOC: Blank Lines of Code, CLOC: Comment Lines of Code, C&LOC: Lines with both Code and Comments , SLOC-L : Logical Source Lines of Code, MVG: McCabe VG Complexity, CWORD: Comment Words, HCLOC: Header Comments, HCWORD: Header Comments Words. The metrics have been described as follows:

- **LOC:** Lines of code represent the total number of lines in the software.
- **SLOP-P:** It is the physical source lines of code and comprises of text of the program's code along with comment lines and blank lines in the code.
- **SLOC-L:** Logical source lines of code calculate the number of statements with specific rules and definitions w.r.t to specific languages.

  SLOC-L = Total Lines of Code - (Comment Lines + Blank Lines) (7)
- **MVG:** McCabe's VG complexity points to the complexity of a program by measuring the number of linearly independent paths in a source code.
- **BLOC:** It stands for blank lines of code.
- **C&SLOC:** It calculates lines consisting of both code and comments.
- **CLOC:** It measures comment lines of code.
- **CWORD:** It measures the count of comment words.
- **HCLOC:** It gives a count for the header comments located just before each class declaration.
- **HCWORD:** It counts the number of words in the header comments.

4.1.3 JarComp Tool for CHANGE Metric

Jarcomp compares Jar documents and Zip records. It's free and cross-platform. In the event that you have two jar files or two zip files, it will demonstrate to you what the distinctions are in the contents. It indicates which files have been included, which have been evacuated, and which are available in both archives. In the event that a document is found in both, it will let you know whether the file has become greater or been diminished in size, or if it's a similar size in both. On the off chance that it's a similar size, a md5 checksum can be made for both files to see whether they're truly similar content or simply a similar size. It is useful in case of comparison of two versions of the same archive to make sure no file was added, deleted or modified mistakenly. It can be also used to make sure that all the modified content was intentionally changed. This tool can be easily handled.

The dependent/output variable used in this study is CHANGE. A CHANGE could be insertion of new line, deletion of a line or simply a modification in any line. We have normalized the value of CHANGE in order to bring it to common scale and allow the comparison of corresponding normalized values for different datasets. ML algorithms also perform better with the normalized values. The formula used to normalize the data is:

$$\boldsymbol{NCHANGE = (max_t - min_t)\left[\frac{CHANGE - min_a}{max_a - min_a}\right] + min_t} \quad (8)$$

Where, max_t – maximum desired target value i.e.10

min_t – minimum desired target value i.e. 0

max_a – maximum value of actual variable

min_a – minimum value of actual variable

4.2 Open Source Software Systems

In this dissertaion, open source software systems have been considered for predicting maintainability as they help in improving industrial relevance and validity of the study by providing generalized results. Further, these software systems and their descriptive statistics have been discussed in the following sub-sections.

4.2.1 Apache Log4j

Apache Log4j [34] is java-based, reliable and fast logging utility framework. Apache Log4j.xml file can be configured as per the needs. Logging is an essential part of software development as it makes debugging quick, maintenance is easy and an application's runtime information s stored in structured manner. It has three components:

- **Loggers:** Capture logging information
- **Appenders:** Publishes the information to preferred destinations
- **Layouts:** Formats the information in different styles.

Apache Log4j.properties file maintains all the configurations. The different logging levels provided by Apache Log4j are:

- ALL
- DEBUG
- ERROR
- FATAL
- INFO
- OFF
- TRACE
- WARN

Version 1.1.2 of Apache Log4j was released on 15th December, 2009 and its source code consisted of 2506 classes. Version 1.1.3 of Apache Log4j was released on 23rd May, 2011 and its source code consisted of 3599 classes. The two versions of the software selected had different number of classes. The number of common classes identified was equal to 2189. CHANGE in a common class is the total count of added, deleted and modified lines. A modification is counted as a deletion and an addition. The ratio of changed to unchanged classes for the above mentioned dataset is depicted in Fig. 4.2. Out of 2189, 886 classes were altered and 1303 were unaltered.

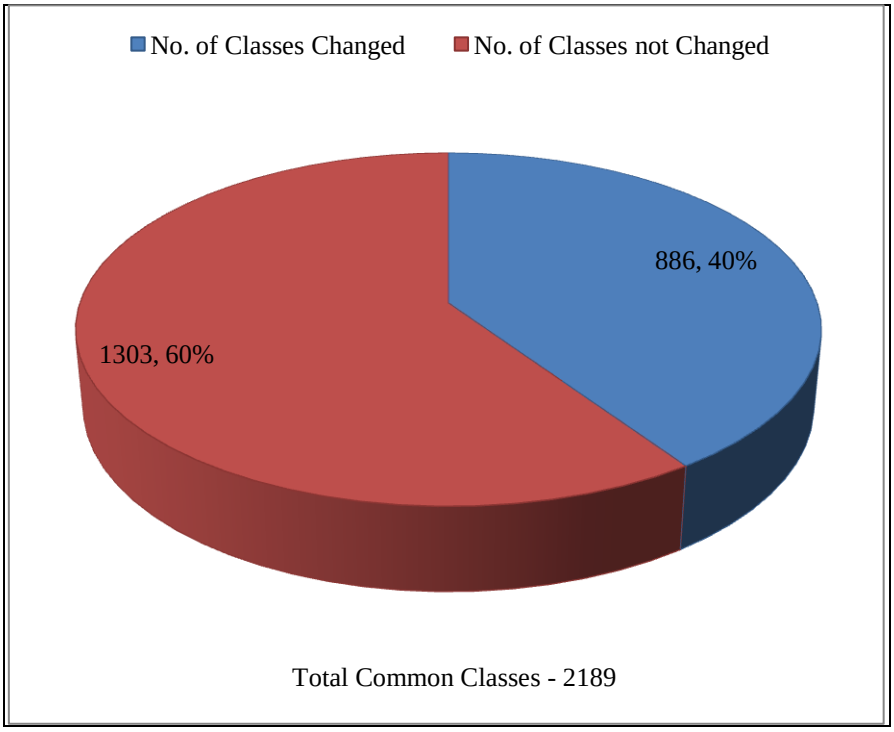

Fig. 4.2 Apache Log4j Class Distribution

The descriptive statistics of Apache Log4j software are tabulated in TABLE 4.1. Further, Fig. 4.3 shows the inter-correlation of different software metrics that are used as input to the training model and Fig. 4.4 shows the correlation of the features with NCHANGE (output variable). Thereafter, the conclusions that have been drawn from these statistics values and graphs have been discussed.

TABLE 4.1 Descriptive Statistics of Apache Log4j Dataset

Metric	Minimum	Maximum	Mean	Median	Standard Deviation	Variance
LOC	1	12548	273.63	153	586.16	343574.90
SLOC-P	1	11468	195.51	90	508.15	258207.90
SLOC-L	0	6753	124.10	63	305.49	93321.67
MVG	0	944	22.53	10	50.62	2562.07
BLOC	0	2443	74.52	39	125.34	15708.74
C&SLOC	0	0	0	0	0	0
CLOC	0	414	3.61	0	14.73	216.98
CWORD	0	552	4.82	0	19.64	385.73
HCLOC	0	0	0	0	0	0
HCWORD	0	0	0	0	0	0
WMC	0	1078	25.16	8	66.36	4403.61
DIT	0	5	0.59	1	0.66	0.43
NOC	0	152	0.36	0	3.69	13.56
CBO	0	243	12.23	9	15.08	227.21
RFC	0	1246	40.23	18	76.81	5898.86
LCOM	0	579426	2387.77	9	20605.74	4.25E+08
Ca	0	243	4.36	2	10.87	117.99
Ce	0	165	8.13	6	10.96	119.97
NPM	0	1077	23.54	6	66.24	4387.51
LCOM3	0	2	1.06	0.97	0.59	0.35
LCO	0	30698	421.75	103	1374.18	1888348
DAM	0	1	0.47	0.37	0.47	0.22
MOA	0	192	1.23	0	8.12	65.85

MFA	0	1	0.03	0	0.15	0.03
CAM	0	1	0.42	0.36	0.26	0.07
IC	0	3	0.05	0	0.22	0.05
CBM	0	20	0.09	0	0.95	0.89
AMC	0	438.50	18.12	11.93	27.53	757.55
NCHANGE	0	100	15.58	14.64	4.84	23.36

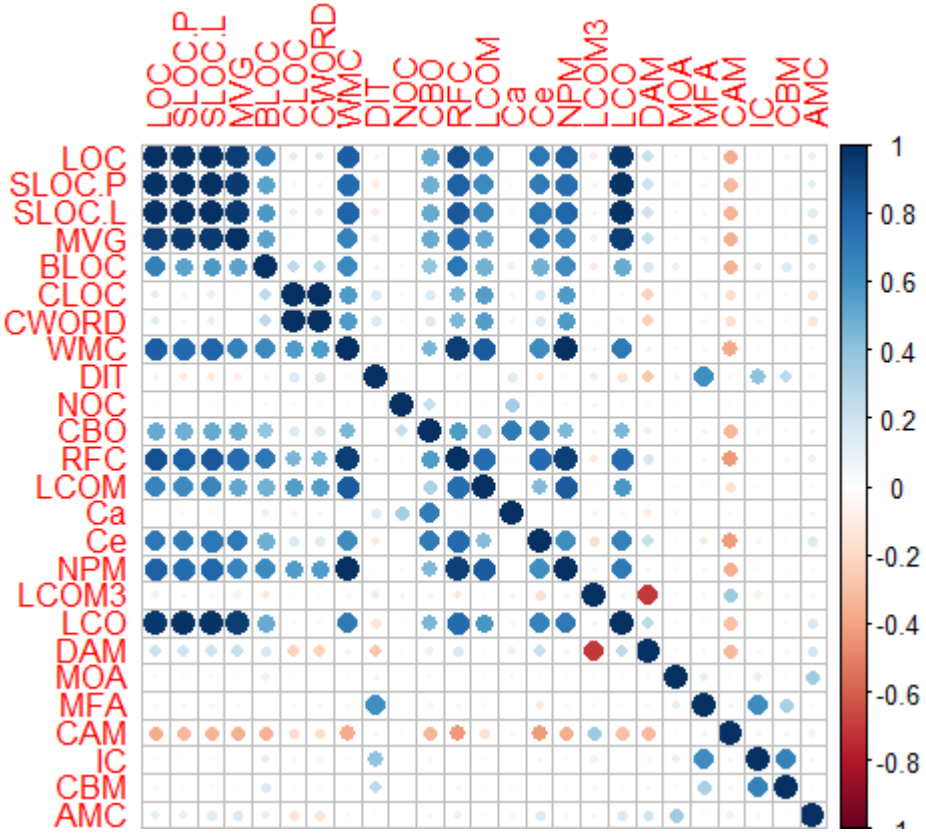

Fig. 4.3 Inter-Correlation of Software Metrics of Apache Log4j Dataset

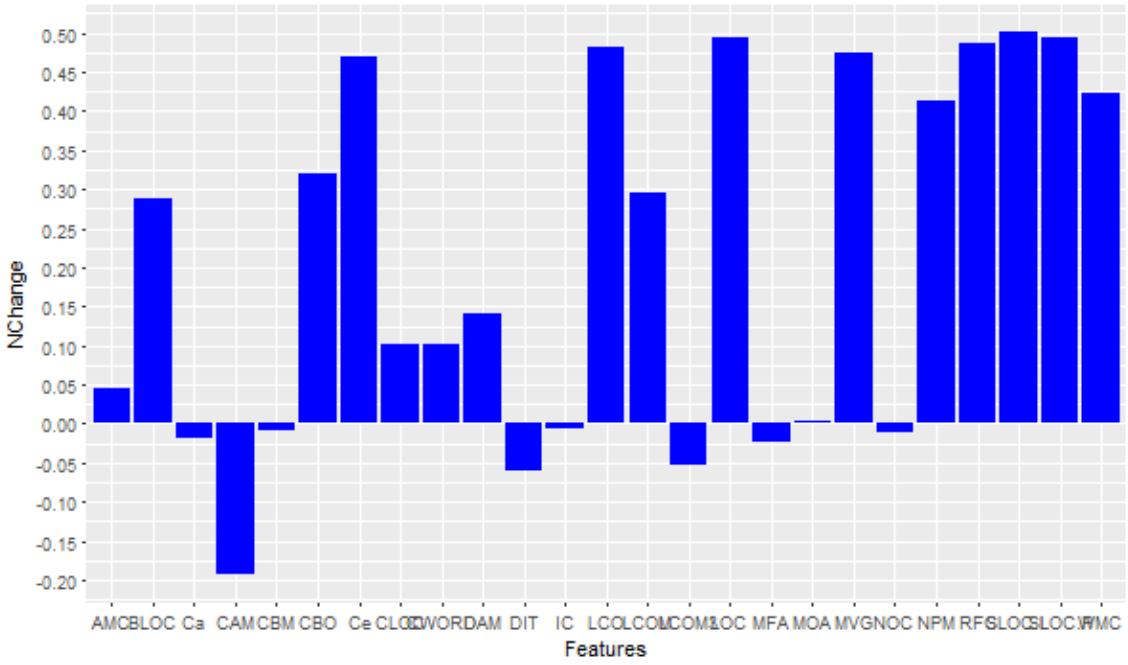

Fig. 4.4 Correlation of Features and NCHANGE of Apache Log4j Dataset

The inferences made based on the descriptive statistics and correlation figures are discussed as follows:

- C&SLOC, HCLOC and HCWORD were removed from the dataset because their values were 0 for all the classes.
- Coupling is a measure of interaction between two classes indicated by Ca, Ce, CBO, IC and CBM metrics. The mean value of CBM is very low, suggesting that the new methods are not coupled with the inherited methods making it a good software design.
- Cohesion is a measure of how well the lines of source code within a module work together. The mean value of LCOM is 2387.77, which is undesirable as high LCOM value implies that the classes are less cohesive and thus, are not efficiently encapsulated.
- The median of NOC is 0 indicating that 50% or more classes do not have child classes. Low value of NOC is undesirable since it does not promote reusability which is one of the key features of object-oriented systems.

- It is preferred to have minimum number of public methods as public means it can be accessed by everyone which leads to security issues. It observed that the mean value of NPM is 23.54.
- The graph in Fig. 4.3 shows how well different metrics are correlated to each other based on which the metrics that are highly correlated can be eliminated as they will affect the value of CHANGE in the same manner.
- Similarly, Fig. 4.4 represents the correlation between input variables (Metrics/Features) and output variable (NCHANGE) i.e. the degree to which input variables are associated with the output variable. It can be negative or positive, though higher absolute value of correlation suggests that a particular metric affects NCHANGE more.

4.2.2 Drumkit

Drumkit [35] is a virtual drumkit which gives you a chance to play percussion sounds by tapping the screen. It is java based mobile application that lets you record the sound and then play later. It is likewise conceivable to play on top of your most recent recording.

Version 1-0.5.0 of Drumkit was released on 18th September, 2016 and its source code consisted of 2376 classes. Version 1-0.6.0 of Drumkit was released on 24th November, 2016 and its source code consisted of 2604 classes. The two versions of the software selected had different number of classes. The number of common classes identified was equal to 1920. CHANGE in a common class is the total count of added, deleted and modified lines. A modification is counted as a deletion and an addition. Out of 1920, 1085 classes were altered and 835 were unaltered. The ratio of changed to unchanged classes for the aforesaid dataset is depicted in Fig. 4.5.

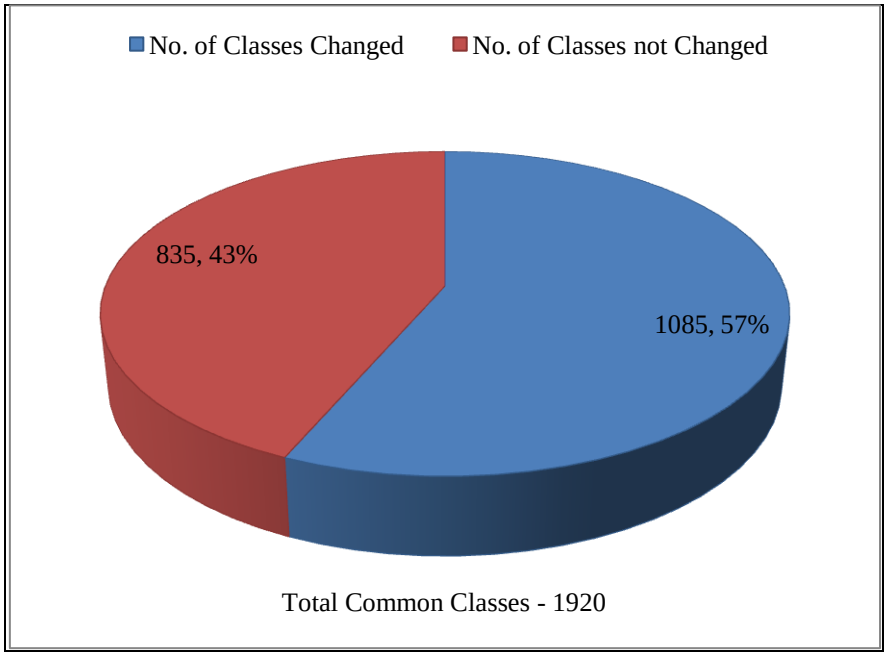

Fig. 4.5 Drumkit Class Distribution

The descriptive statistics of Drumkit software are tabulated in TABLE 4.2. Further, Fig. 4.6 shows the inter-correlation of different software metrics that are used as input to the training model and Fig. 4.7 shows the correlation of the features with NCHANGE (output variable). Further, the conclusions that have been drawn from these statistics values and graphs have been discussed.

TABLE 4.2 Descriptive Statistics of Drumkit Dataset

Metric	Minimum	Maximum	Mean	Median	Standard Deviation	Variance
LOC	1	3120	113.25	67	166.44	27703.33
SLOC-P	1	1788	71.63	37	109.35	11957.01
SLOC-L	0	1253	49.03	25	74.81	5596.81
MVG	0	301	10.54	4	21.18	448.44
BLOC	0	1670	41.39	29	63.39	4017.72
C&SLOC	0	0	0	0	0	0
CLOC	0	93	0.23	0	2.79	7.79

CWORD	0	441	0.65	0	11.87	140.86
HCLOC	0	0	0	0	0	0
HCWORD	0	0	0	0	0	0
WMC	0	198	9.50	5	13.66	186.56
DIT	0	4	0.57	1	0.63	0.40
NOC	0	71	0.52	0	3.02	9.11
CBO	0	531	11.75	7	25.24	636.98
RFC	0	444	23.13	14	30.23	913.68
LCOM	0	19407	111.87	4	766.46	587454.72
Ca	0	529	5.83	2	24.05	578.46
Ce	0	108	6.42	4	7.79	60.76
NPM	0	184	7.87	4	12.30	151.41
LCOM3	0	2	1.12	0.85	0.72	0.51
LCO	0	4921	134	59	239.47	57343.67
DAM	0	1	0.60	1	0.48	0.23
MOA	0	26	0.66	0	1.47	2.16
MFA	0	1	0.03	0	0.15	0.02
CAM	0	1	0.49	0.44	0.27	0.07
IC	0	2	0.02	0	0.14	0.02
CBM	0	7	0.03	0	0.26	0.07
AMC	0	158.67	11.26	8.30	12.51	156.53
NCHANGE	0	100	38.25	36.44	5.85	34.18

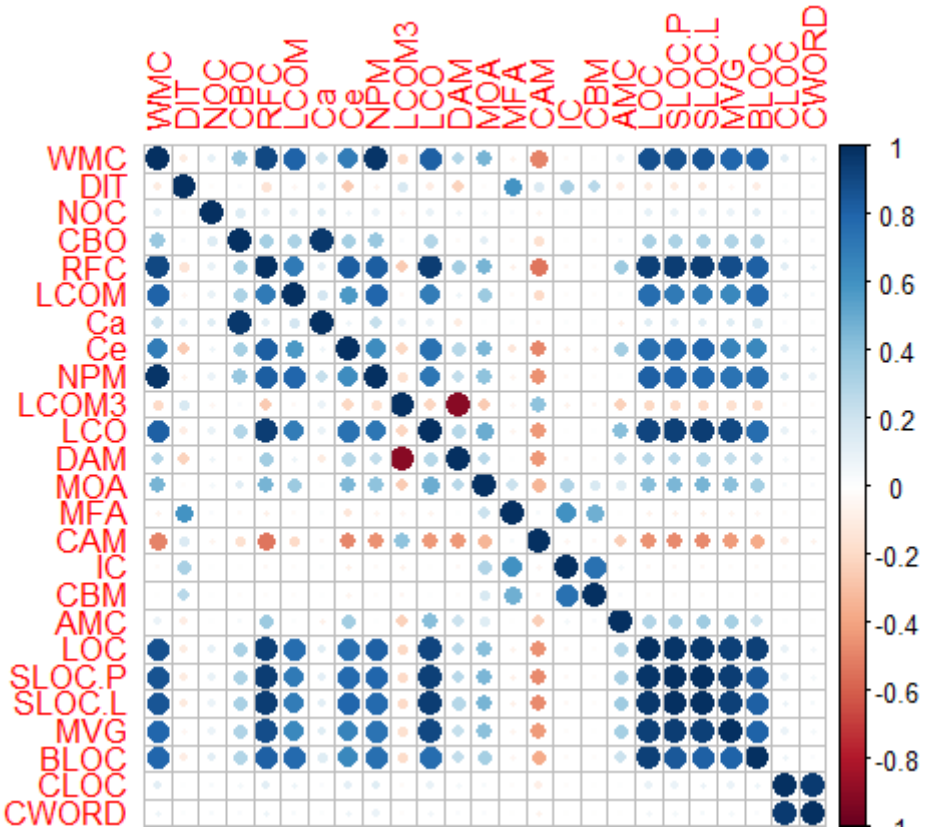

Fig. 4.6 Inter-Correlation of Software Metrics of Drumkit Dataset

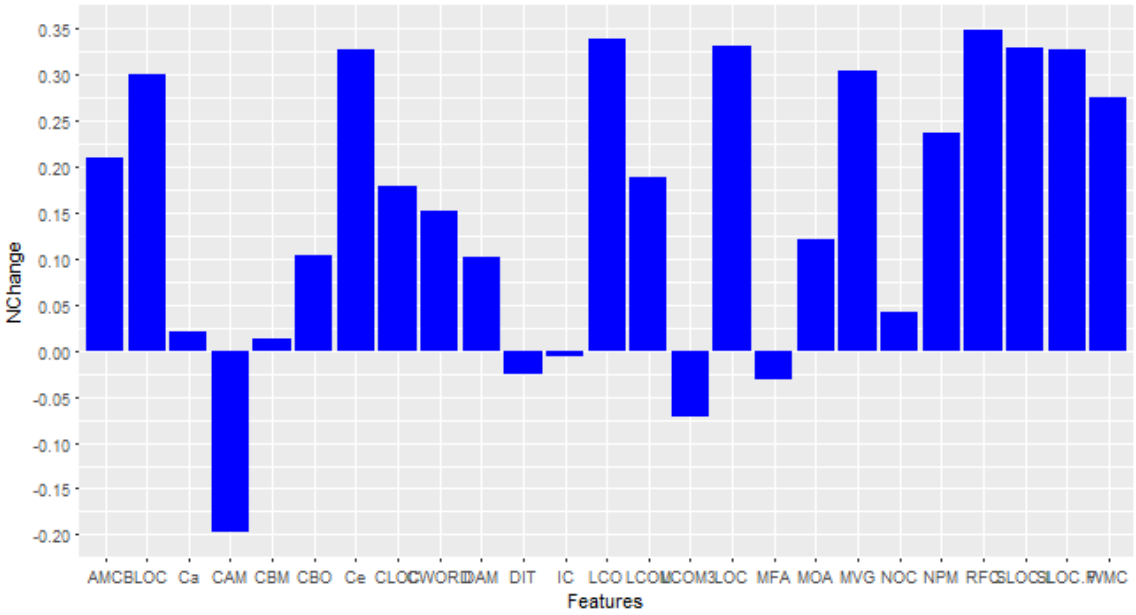

Fig. 4.7 Correlation of Features and NCHANGE of Drumkit Dataset

The following inferences have been made based on the descriptive statistics and correlation figures above:

- C&SLOC, HCLOC and HCWORD were removed from the dataset because their values were 0 for all the classes.
- Coupling is a measure of interaction between two classes indicated by Ca, Ce, CBO, IC and CBM metrics. The mean value of CBM is very low i.e. 0.03, suggesting that the new methods are not coupled with the inherited methods making it a good software design.
- Cohesion is a measure of how well the lines of source code within a module work together. The mean value of LCOM is 111.87, which is desirable as low LCOM value implies that the classes are more cohesive and thus, are efficiently encapsulated.
- The median of NOC is 0 indicating that 50% or more classes do not have child classes. Low value of NOC is undesirable since it does not promote reusability which is one of the key features of object-oriented systems.
- It is preferred to have minimum number of public methods as public means it can be accessed by everyone which leads to security issues. It observed that the mean value of NPM is 7.87 which is desirable.
- The graph in Fig. 4.6 shows how well different metrics are correlated to each other based on which the metrics that are highly correlated can be eliminated as they will affect the value of CHANGE in the same manner.
- Similarly, Fig. 4.7 represents the correlation between input variables (Metrics/Features) and output variable (NCHANGE) i.e. the degree to which input variables are associated with the output variable. It can be negative or positive, though higher absolute value of correlation suggests that a particular metric affects NCHANGE more.

4.3 Feature Selection Algorithms

In the study performed, seven FS algorithms have been used which are described below:

4.3.1 CFS

CFS algorithm finds feature subset by using correlation and entropy readings for discrete and continuous data. This algorithm chooses subsets of features having high correlation with the class and low inter-correlation. It uses best first search as the searching criteria [36].

4.3.2 Consistency

Consistency FS algorithm can be used for both continuous and discrete data for feature reduction. It uses best first search to shortlist features from the attribute space on the basis of consistency. The highest consistent features are obtained as output. A FS technique is said to be conflicting if minimum two instances with same feature values have dissimilar output [2].

4.3.3 Boruta

It is one of the quick wrapper methods of FS that finds the relevance of a feature by creating shadow features. Shadow features are the shuffled copies of all features. It uses mean decrease accuracy to evaluate the importance and At every step the importance of real feature is matched against shadow feature and unimportant features are removed. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs [37].

4.3.4 Relief

It is an instance-based FS algorithm which finds distance between instances for evaluating weights of continuous and discrete attributes. The algorithm samples instances and finds their nearest hits and misses. Thus, on the basis of results the weights are calculated. It identifies features that are statistically significant to the target variable and the output is a data frame of weights and features [38].

4.3.5 OneR

It measures weights of attributes based on simple association rules having only one attribute in the condition part. The classifier used to find weights by this algorithm is also OneR. A simple rule is created for each attribute based on that attribute and then its error rate is calculated. The output us a data frame with a worth of features followed by their names. [38].

4.3.6 Random Forest

In random forest the weights of attributes are calculated using random forest algorithm. Random forest uses decision trees for FS by splitting data set into two based on similar response values and further, variance decrease is factored while training the tree [38].

4.3.7 Recursive Feature Elimination

Recursive Feature Elimination works on the principle of initial inclusion of all features and gradual exclusion based on weighted value of a feature. It uses cross validation to increase the probability of relevant features being included. Each iteration removes few features least affecting the outcome till features which have maximum impact on output are left.

4.4 Machine Learning Algorithms

The various ML algorithms used are described below:

4.4.1 Linear Regression

It finds the relationship between the input variables and output variable by fitting the data along a straight line called the regression line. The input variables can be continuous or discrete and the output variable is continuous. The most common method to obtain regression line is Least Square Method [39].

$$D = a + b * I + E \tag{9}$$

Where, a - Intercept

b - Slope of line

E – Error

4.4.2 General Regression Neural Network

GRNN helps to construct models that can recognize complex patterns and non-linear forecasting task show very good results when using these models [40]. They work similar to the human neural network by taking the inputs through dendrites and based on these inputs they produce the outputs. The simplest neural network has one perceptron and multiple perceptron's called hidden layers can be added based on requirements.

4.4.3 Decision Tree

Decision trees perform binary splits on the recursive predictors. The predictors are partitioned into 'M' regions R_1, R_2, ……., R_M and the response 'y' is modeled as the average for a region with

$$\hat{f}(x) = \sum_{m=1}^{M} \hat{c}_m \, I(x \, \epsilon \, R_m) \tag{10}$$

Where, $$\hat{c}_m = avg(y_i | x_i \, \epsilon \, R_m) \tag{11}$$

is the average ‘y’ value for the region [41].

4.4.4 Cubist

It is a rule based model in which a tree with terminal nodes having linear regression model is manufactured. In each progression of the tree intermediate linear models are available. The tree is combined to a set of guidelines, which are fundamentally ways from the top to the base of the tree. Rules are reduced utilizing pruning or joined for rearrangements [37].

4.4.5 Ridge Regression with Variable Selection

It works in the similar manner as least squares, but also uses a tuning parameter to shrink the anticipated coefficients towards zero. It is decided by the size of the tuning parameter using cross validation. Variable selection is done using Forward Greedy, Backward Greedy and Adaptive Forward-Backward Greedy (FoBa) methods [42].

4.4.6 LASSO

LASSO stands for Least Absolute Shrinkage and Selection Operator. It ignores the features having zero coefficient leading to sparse solutions, hence improving the performance when the number of features are very high. It reduces the coefficient of other features to zero by randomly selecting a feature from set of highly co-related features. LASSO typically performs L1 regularization wherein, it adds a factor of sum of absolute value of coefficients in the optimization objective [37]. However, this does not perform as well as ridge regression.

4.4.7 Elastic Net

Elastic Net typically works best when multiple features are correlated. It is the combination of both Ridge Regression and LASSO. Elastic Net picks two features at random unlike LASSO that picks only one. It can sometime degrade double shrinkage. The trade-off between Ridge and LASSO allows this algorithm to incorporate stability under rotation from Ridge [43].

4.4.8 Random Forest

This prediction algorithm implements Breiman's random forest for classification and regression. Random forest is a collection of decision trees that classifies new objects. For accessing proximities among data points, this algorithm can be used in unsupervised mode also [44].

4.4.9 Principal Component Analysis

Principal component analysis find the principal predictors and then use these predictors for the construction of linear regression model fitted using the least square method. It assumes that the direction in which the predictors show the most variation are the exact directions associated with the response variable. The advantages of using this FS algorithm are dimensionality reduction, avoidance of multi-collinearity between predictors and over-fitting [37].

4.5 Prediction Accuracy Measures

Prediction accuracy measures are the means of evaluating the quality of the model built and its performance level. The different evaluation parameters used in this study to find the best model among different FS and ML combinations are:

- **Mean Absolute Error (MAE)**

  MAE calculates the average of the absolute error i.e. the difference between predicted and the actual values**.** The equation below shows the formula to calculate MAE:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | P_i - A_i | \tag{12}$$

  Where, N – Total no. of instances

  P_i – Predicted value

  A_i – Actual value

- **Root Mean Square Error (RMSE)**

  This measure typically indicates how the data is spread around the regression line i.e. the standard deviation of errors and is calculated using equation:

$$RMSE = \sqrt[2]{\frac{1}{N}\sum_{i=1}^{N}(P_i - A_i)^2} \tag{13}$$

- **Coefficient of Correlation (r)**

  Correlation measures the strength of association between two variables. Its value varies from -1 to 1 where 1 and -1 define the perfect relationship and 0

indicates weak relationship. The negative sign indicates that the two variables are inversely related. Pearson Coefficient of correlation is:

$$r = \frac{\sum(P_i - \bar{P})(A_i - \bar{A})}{\sqrt[2]{\sum(P_i - \bar{P})^2(A_i - \bar{A})^2}} \quad (14)$$

- **Accuracy**

It calculates how close the predicted value is to the actual value.

$$Accuracy = \left(\frac{1}{N}\sum_{j=1}^{N}|P_j - A_j|\right) * 100 \quad (15)$$

Where, $|P_j - A_j| \leq 1$ and 1 is the fault tolerance that can be changed by the user as per his needs.

CHAPTER – 5 IMPLEMENTATION & RESULTS

The first step as specified by the research methodology has been to identify features i.e. set of metrics that can best predict the value of CHANGE to enhance the structural quality of the software. Further, ML algorithms have been applied to predict software maintenance and different combinations of FS and ML techniques are analyzed.

5.1 Comparison of Different Feature Selection Algorithms

Different FS algorithms provide different subset of features with varying sizes based on their underlying working. TABLE 5.1 and 5.2 show the set of most significant metrics obtained by applying the seven FS algorithms – CFS, Consistency, Boruta, Relief, OneR, Random Forest and Recursive Feature Elimination for Apache Log4j and Drumkit dataset respectively.

- According to CFS algorithm RFC, Ce, LCO and SLOC-L are the most significant metrics for Apache Log4jwhereas after applying CFS for Drumkit DIT, RFC, LCO, DAM, SLOC-L and BLOC are found to be more significant than others.
- Consistency FS algorithm gave LOC, SLOC-P, SLOC-L, MVG, WMC, DIT, CBO, RFC, LCOM, Ca, Ce, NPM, LCOM3, LCO and DAM as the most relevant metrics for Apache Log4j and WMC, DIT, CBO, RFC, LCOM, Ce, NPM, LCOM3, LCO, DAM, MOA, CAM, AMC, LOC, SLOC-P and SLOC-L for Drumkit.
- Boruta FS technique shortlisted LOC, SLOC-P, SLOC-L, MVG, WMC, RFC, LCOM, Ca, Ce, NPM, LCOM3, LCO, DAM, MFA, CAM and AMC for Apache Log4j and WMC, RFC, LCOM, Ca, Ce, BLOC, LCO, DAM, SLOC-L and MVG for Drumkit dataset.
- WMC, CBO, NPM, SLOC-P, NOC, Ca, LCOM, MFA, IC, CBM, DAM, LOC and MOA are most relevant metrics obtained by Relief FS algorithm for Apache Log4j and WMC, NPM, DIT, BLOC, MOA, LOC, NOC, LCOM3, MVG, RFC, SLOC-P, LCOM, Ca and Ce for Drumkit dataset.

- Similarly, CBM, CLOC, CWORD, MFA, MOA, CBO, DAM, DIT, Ce, WMC, Ca, SLOC-L, NPM, SLOC-P and LCOM are found to be more important for OneR FS algorithm in case of Apache Log4j dataset and DIT, NOC, IC, CBM, MFA, Ce, SLOC-P, MOA, DAM, LOC, LCOM and SLOC-L for Drumkit dataset.
- According to Random Forest FS algorithm CLOC, CWORD, MVG, RFC, BLOC, LOC, LCO, SLOC-P, SLOC-L, LCOM3, NPM, DAM, Ce, WMC and MFA and Ce, NOC, LCOM3, BLOC, AMC, MVG, RFC, LCO, NPM, SLOC-P, LCOM, SLOC-L, CAM, CBO and LOC are the most important metrics for Apache Log4jand Drumkit respectively.
- Recursive Feature Elimination when applied for Apache Log4j gave RFC, BLOC, MVG, LOC, SLOC-P, NPM, LCO, SLOC-L, LCOM3, Ce, WMC, DAM, LCOM and CBO and for Drumkit Ce, NOC, LCOM3, BLOC, AMC, MVG, RFC, LCO, NPM, SLOC-P, LCOM, SLOC-L, CAM, CBO and LOC
- It has been observed that SLOC-L, SLOC-P, Ce, WMC, NPM and DAM have been selected as one of the significant metrics by six out of seven FS techniques for Apache Log4j dataset. Additionally, LCO, RFC, LOC and LCOM have been selected by five algorithms.
- Similarly, RFC, LCOM and Ce have been shortlisted by six out of seven FS algorithms for Drumkit and LCO, SLOC-L, LOC, SLOC-L and SLOC-P by five FS methods.
- Thus, the key metrics identified by FS techniques for further prediction are LCO, RFC, SLOC-L, SLOC-P, LOC and LCOM.

TABLE 5.1 Metrics Subset Obtained by FS Algorithms for Apache Log4jDataset

CFS	Consistency	Boruta	Relief	OneR	Random Forest	Recursive Feature Elimination
RFC	LOC	LOC	WMC	CBM	CWORD	RFC
Ce	SLOC-P	SLOC-P	CBO	CLOC	CLOC	BLOC
LCO	SLOC-L	SLOC-L	NPM	CWORD	MVG	MVG
SLOC-L	MVG	MVG	SLOC-P	MFA	RFC	LOC
	WMC	WMC	NOC	MOA	BLOC	SLOC-P
	DIT	RFC	Ca	CBO	LOC	NPM

	CBO	LCOM	LCOM	DAM	LCO	LCO
	RFC	Ca	MFA	DIT	SLOC-P	SLOC-L
	LCOM	Ce	IC	Ce	SLOC-L	LCOM3
	Ca	NPM	CBM	WMC	LCOM3	Ce
	Ce	LCOM3	DAM	Ca	NPM	WMC
	NPM	LCO	LOC	SLOC-L	DAM	DAM
	LCOM3	DAM	MOA	NPM	Ce	LCOM
	LCO	MFA		SLOC-P	WMC	CBO
	DAM	CAM		LCOM	MFA	
		AMC				

TABLE 5.2 Metrics Subset Obtained by FS Algorithms for Drumkit Dataset

CFS	Consistency	Boruta	Relief	OneR	Random Forest	Recursive Feature Elimination
DIT	WMC	WMC	WMC	DIT	Ce	Ce
RFC	DIT	RFC	NPM	NOC	NOC	NOC
LCO	CBO	LCOM	DIT	IC	LCOM3	LCOM3
DAM	RFC	Ca	BLOC	CBM	BLOC	BLOC
SLOC-L	LCOM	Ce	MOA	MFA	AMC	AMC
BLOC	Ce	BLOC	LOC	Ce	MVG	MVG
	NPM	LCO	NOC	SLOC-P	RFC	RFC
	LCOM3	DAM	LCOM3	MOA	LCO	LCO
	LCO	SLOC-L	MVG	DAM	NPM	NPM
	DAM	MVG	RFC	LOC	SLOC-P	SLOC-P
	MOA		SLOC-P	LCOM	LCOM	LCOM
	CAM		LCOM	SLOC-L	SLOC-L	SLOC-L
	AMC		Ca		CAM	CAM
	LOC		Ce		CBO	CBO
	SLOC-P				LOC	LOC
	SLOC-L					

The next step is to identify the most efficient FS algorithm on the basis of different prediction accuracy parameters. Further, nine ML algorithms were applied on the original datasets and those obtained by the seven FS methods. The evaluation results for each ML algorithm with the original dataset and different FS techniques are discussed further and respective graphs for prediction accuracy measures are depicted as follows:

- The evaluation results for Linear Regression ML algorithm are tabulated in TABLE 5.3 and the prediction accuracy parameters r, MAE, RMSE and Accuracy are depicted in Fig. 5.1, 5.2, 5.3 and 5.4 respectively. Linear Regression shows improvement of 0.14 in r, 0.17 in MAE, 0.16 in RMSE and 9.20% in Accuracy for Apache Log4j whereas 0.13 in r, 0.26 in MAE, 0.27 in RMSE and 9.95% in Accuracy for Drumkit with Relief FS method.
- General Regression with Neural Network gives better results after FS using CFS technique. An increase of 0.37, decrease of 0.11 and 0.12 and increase of 5.99% in r, MAE, RMSE and Accuracy respectively was found for Apache Log4j and 0.08, 0.11and 13.22% in r, MAE and Accuracy respectively for Drumkit. The results are shown in TABLE 5.4 and Fig. 5.5, 5.6, 5.7 and 5.8.
- Decision Tree prediction results are summarized in TABLE 5.5 and the error detection parameters are plotted in Fig. 5.9, 5.10, 5.11 and 5.12. For Apache Log4j Decision Tree shows an increase of 0.16 in r, 46.66% and 18.18% fall in MAE and RMSE and 10.05% increase in Accuracy was seen with Recursive Feature Elimination. Similarly, for Drumkit and increase of 0.15 in r, 77.36% and 69.62% fall in MAE and RMSE and 7.2% increase in Accuracy was seen with Consistency.
- TABLE 5.6 showcases the outcome of Cubist algorithm with different FS methods. The graphs for the same are shown in Fig. 5.13, 5.14, 5.15 and 5.16. Cubist gives better results after FS by OneR method for both the datasets. An increase of 22.00% and 3.57% in r, decrease of 0.05 and 0.02 in MAE, fall of 0.02 and 0.04 in RMSE and rise of 2.24% and 5.7% in Accuracy was found for Apache Log4j and Drumkit correspondingly.
- For both Apache Log4j and Drumkit, the results of Ridge Regression with Variable respectively are reflected in TABLE 5.7. The values for r, MAE,

RMSE and Accuracy are graphed in Fig. 5.17, 5.18, 5.19 and 5.20 respectively. The combination of Consistency FS method and Ridge Regression with Variable Selection ML algorithm for Apache Log4j and Relief FS method and Ridge Regression with Variable Selection for Drumkit dataset gives enhanced results. A hike of 0.13 in r, fall of 0.03 and 0.04 in MAE and RMSE respectively and increase of 8.61% in Accuracy for Apache Log4j and similarly, 0.15 in r, 0.04 and 0.09 in MAE and RMSE and 5.12% in Accuracy for Drumkit was observed.

- The prediction results for LASSO with various FS methods are tabulated in TABLE 5.8 and the values of r are shown in Fig. 5.21, MAE in Fig. 5.22, RMSE in Fig. 5.23 and Accuracy in Fig. 5.24. LASSO gave superior results with Consistency FS method for both datasets as follows: rise of 0.0.9 and 0.15 in r and 7.58% and 8.06% in Accuracy and fall of 20.00% and 33.33% in MAE and 35.42% and 29.31% in RMSE for Apache Log4j and Drumkit respectively.
- TABLE 5.9 summarizes the outcome of Elastic Net and Fig. 5.25, 5.26, 5.27 and 5.28 plot r, MAE, RMSE and Accuracy correspondingly. Elastic Net gave best results with Recursive Feature Elimination algorithm for Apache Log4j and with Consistency for Drumkit. For Apache Log4j MAE by 0.05, RMSE by 0.34 and Accuracy 23.29% and for Drumkit r by 0.04, MAE by 0.17, RMSE by 0.31 and Accuracy by 2.14%.
- The results for Random Forest algorithm are shown in TABLE 5.10 and in Fig. 5.29, 5.30, 5.31 and 5.32. For Apache Log4j Random Forest with OneR gave improved results as compared to with original dataset. r and Accuracy increased by 0.03 and 14.13% and MAE and RMSE decreased by 0.10 and 0.04. Similarly, for Drumkit Random Forest with OneR gave improved results as follows: r by 0.03, Accuracy by 6.82%, MAE by 0.13 and RMSE by 0.13.
- Evaluation results for Principal Component Analysis ML algorithm are showcased in TABLE 5.11 and the measures for error detection are depicted in Fig. 5.33, 5.34, 5.35 and 5.36. The combination of Consistency FS technique with Principal Component Analysis ML algorithm for Apache Log4j and OneR FS technique with Principal Component Analysis for Drumkit show superior results in comparison to prediction results of original

dataset without any feature reduction. MAE and RMSE declined by 7.14% and 34.04% and 51.85% and 44.23% and Accuracy enhanced by 3.65% and 4.16% for Apache Log4j and Drumkit correspondingly.

TABLE 5.3 Evaluation Results for Linear Regression with Different FS Algorithms

Linear Regression								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.44	0.35	0.50	69.45	0.33	0.60	0.62	75.78
CFS	0.50	0.26	0.35	71.50	0.36	0.48	0.53	78.50
Consistency	0.52	0.20	0.32	75.23	0.43	0.38	0.23	82.68
Boruta	0.53	0.38	0.47	71.01	0.40	0.50	0.65	81.11
Relief	0.58	0.18	0.34	78.65	0.46	0.34	0.35	85.73
OneR	0.49	0.26	0.44	69.50	0.42	0.50	0.59	79.17
Random Forest	0.57	0.38	0.46	71.06	0.36	0.50	0.58	82.06
Recursive Feature Elimination	0.53	0.30	0.46	70.05	0.40	0.54	0.63	80.60

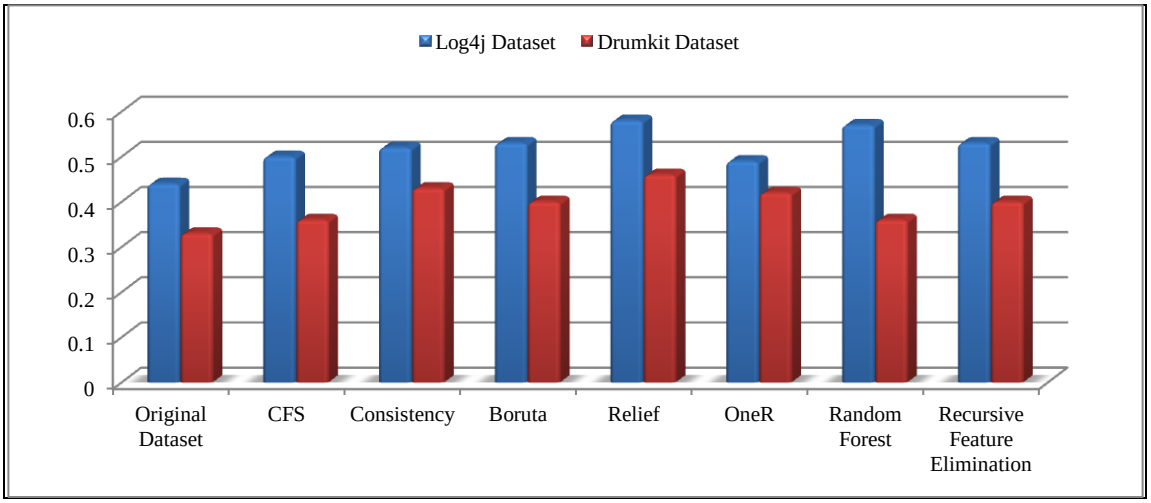

Fig. 5.1 Coefficient of Correlation of Various FS Algorithms with Linear Regression

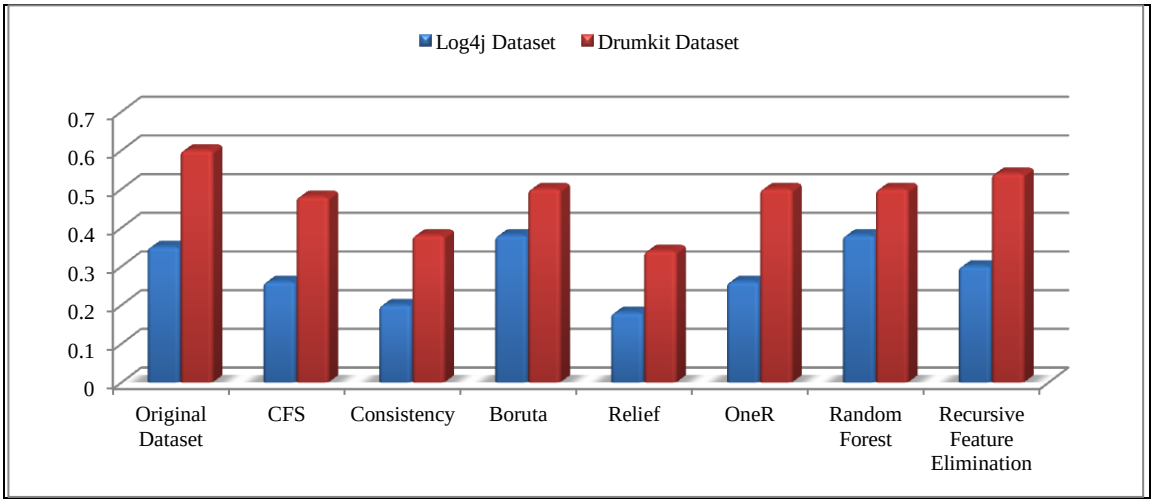

Fig. 5.2 MAE of Various FS Algorithms with Linear Regression

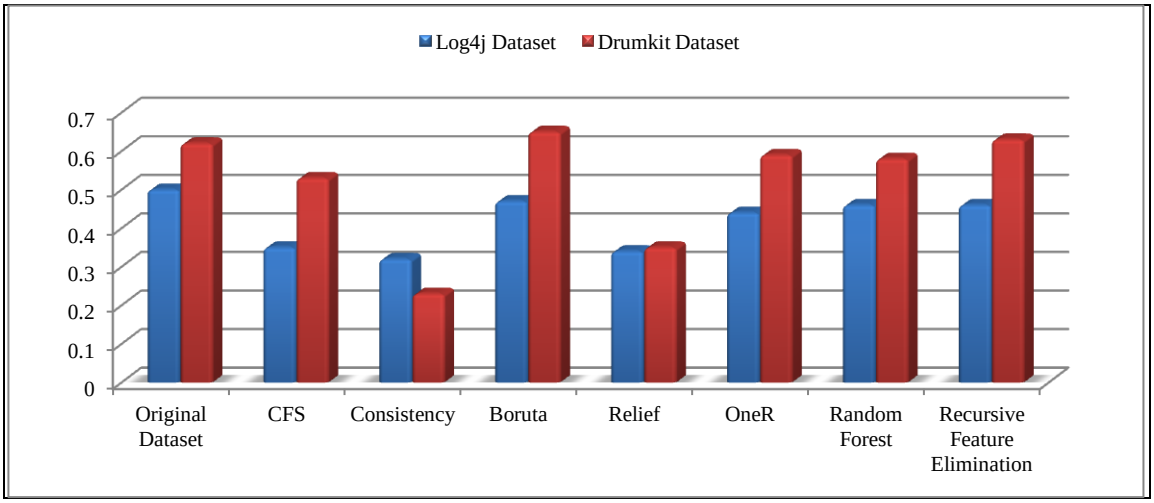

Fig. 5.3 RMSE of Various FS Algorithms with Linear Regression

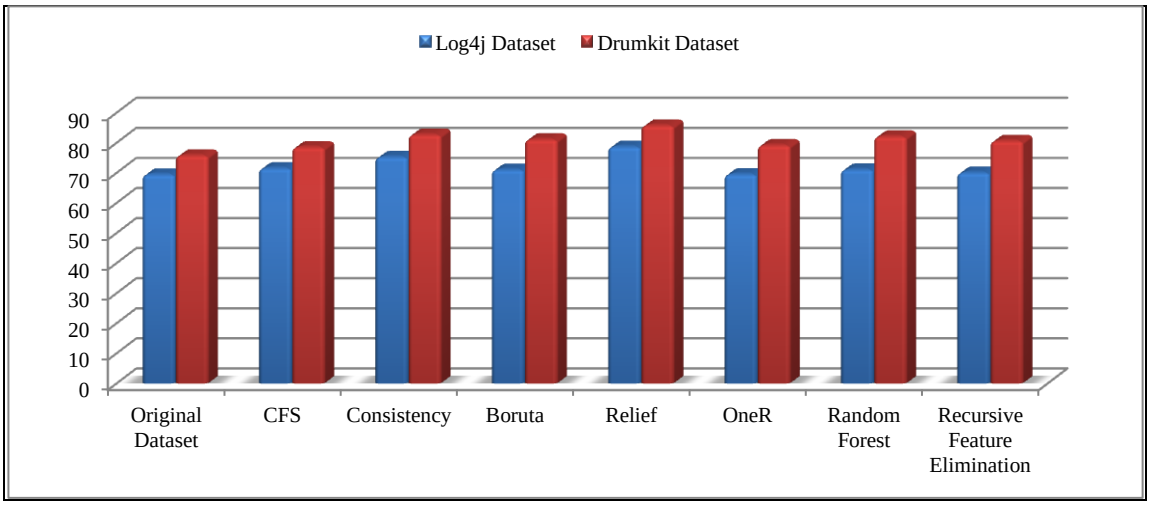

Fig. 5.4 Accuracy of Various FS Algorithms with Linear Regression

TABLE 5.4 Evaluation Results for General Regression Neural Network with Different FS Algorithms

General Regression Neural Network								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.29	0.37	0.50	70.61	0.27	0.60	0.94	72.77
CFS	0.66	0.26	0.38	82.60	0.35	0.19	0.60	85.99
Consistency	0.48	0.34	0.40	72.60	0.26	0.26	0.67	80.15
Boruta	0.40	0.30	0.44	73.11	0.16	0.37	0.33	81.01
Relief	0.48	0.29	0.38	78.99	0.20	0.28	0.65	79.94
OneR	0.53	0.32	0.52	71.13	0.36	0.35	0.40	79.13
Random Forest	0.50	0.30	0.47	74.79	0.27	0.54	0.60	74.79
Recursive Feature Elimination	0.47	0.30	0.28	74.33	0.25	0.60	0.85	74.33

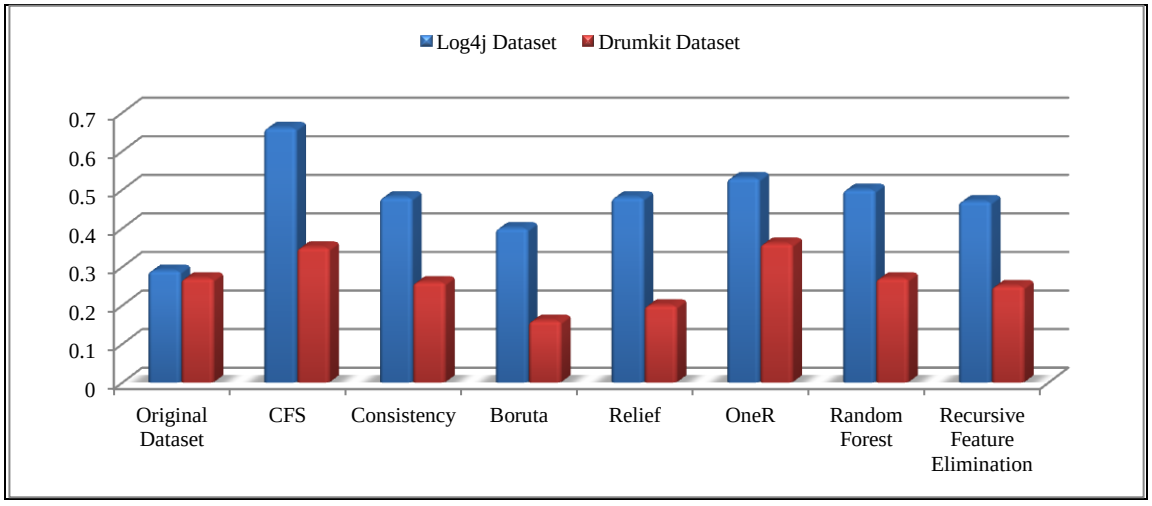

Fig. 5.5 Coefficient of Correlation of Various FS Algorithms with General Regression Neural Network

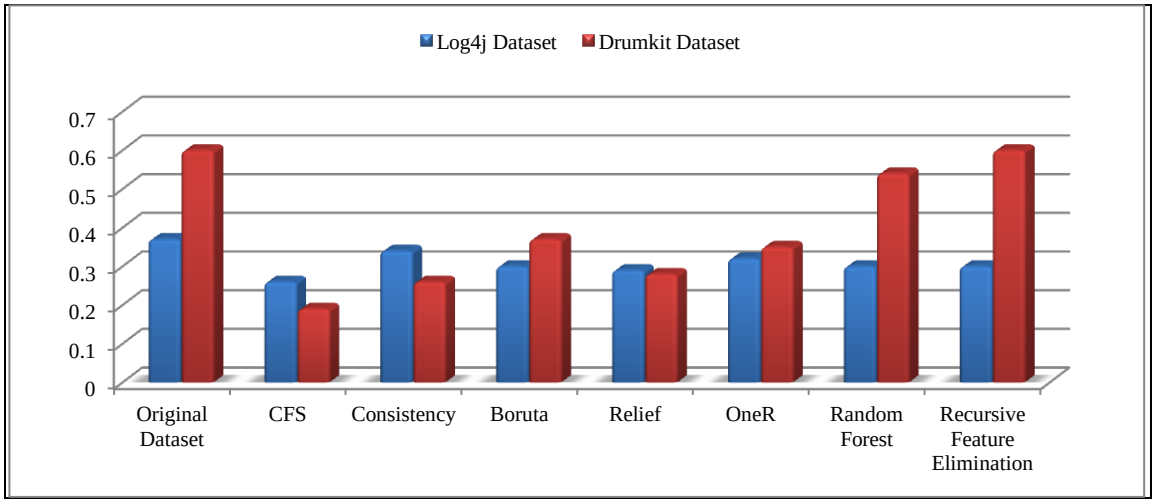

Fig. 5.6 MAE of Various FS Algorithms with General Regression Neural Network

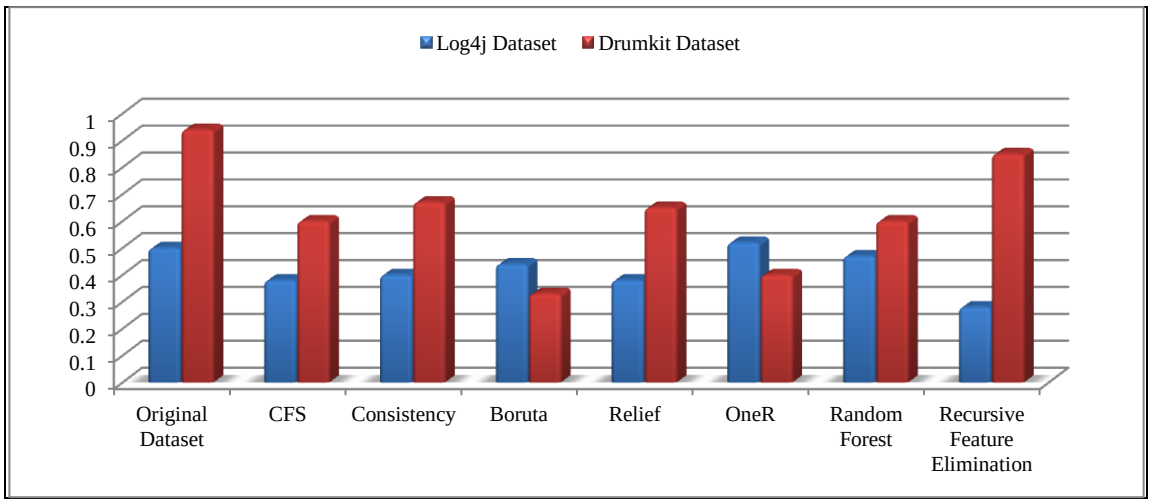

Fig. 5.7 RMSE of Various FS Algorithms with General Regression Neural Network

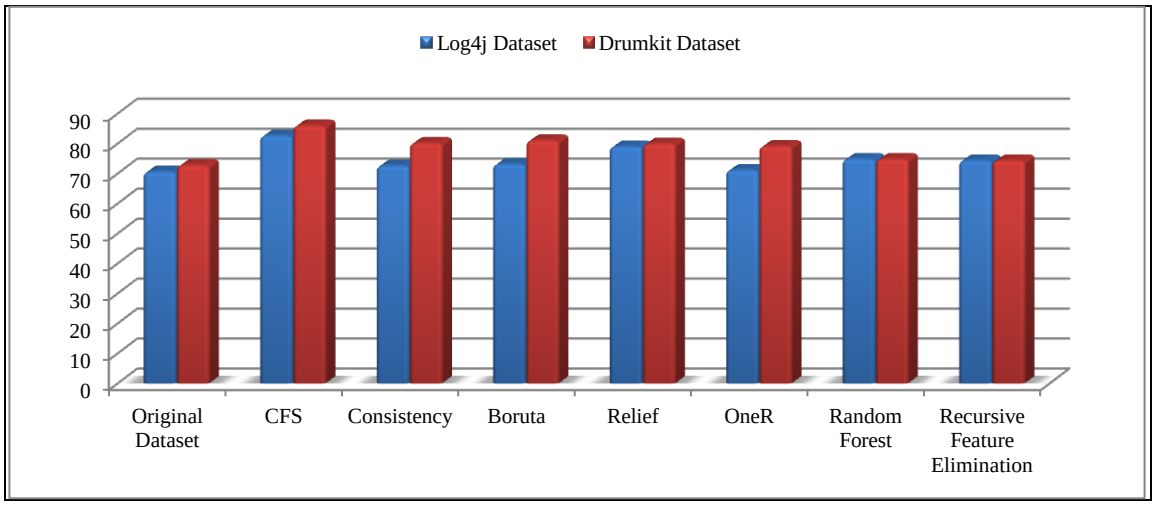

Fig. 5.8 Accuracy of Various FS Algorithms with General Regression Neural Network

TABLE 5.5 Evaluation Results for Decision Tree with Different FS Algorithms

Decision Tree								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.48	0.45	0.44	72.65	0.27	0.53	0.79	70.88
CFS	0.58	0.28	0.38	82.00	0.26	0.53	0.65	71.50
Consistency	0.56	0.25	0.28	80.23	0.42	0.12	0.24	78.08
Boruta	0.65	0.26	0.35	81.64	0.32	0.52	0.66	68.64
Relief	0.54	0.14	0.40	85.93	0.40	0.14	0.27	75.91
OneR	0.55	0.26	0.36	84.53	0.39	0.52	0.68	68.44
Random Forest	0.58	0.26	0.49	84.07	0.28	0.52	0.57	68.07
Recursive Feature Elimination	0.64	0.24	0.36	82.70	0.32	0.48	0.63	75.70

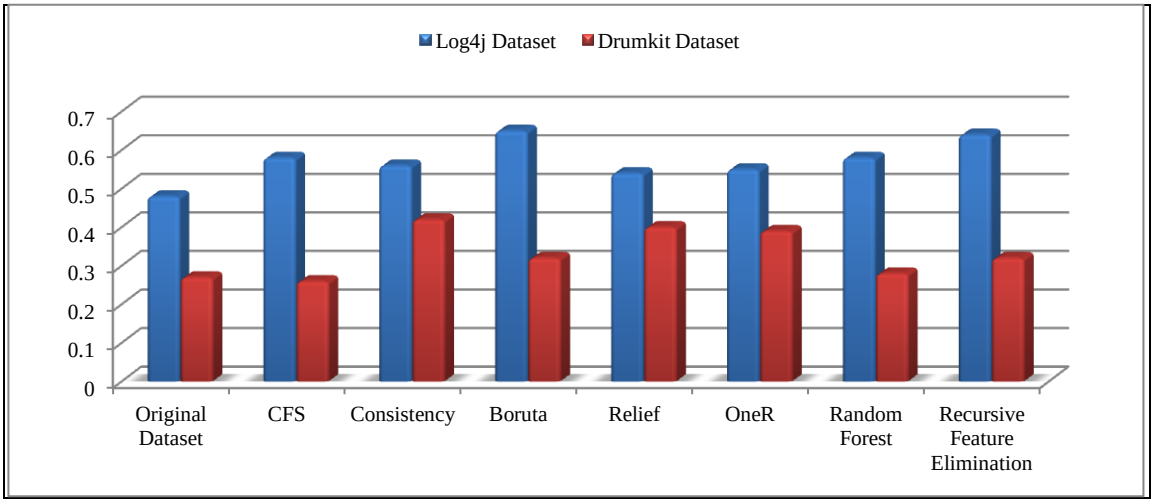

Fig. 5.9 Coefficient of Correlation of Various FS Algorithms with Decision Tree

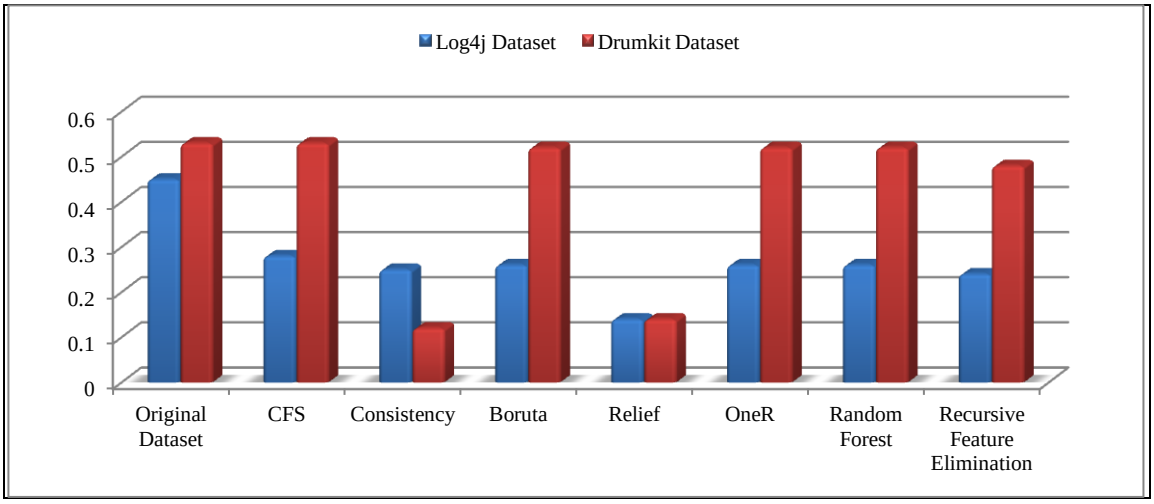

Fig. 5.10 MAE of Various FS Algorithms with Decision Tree

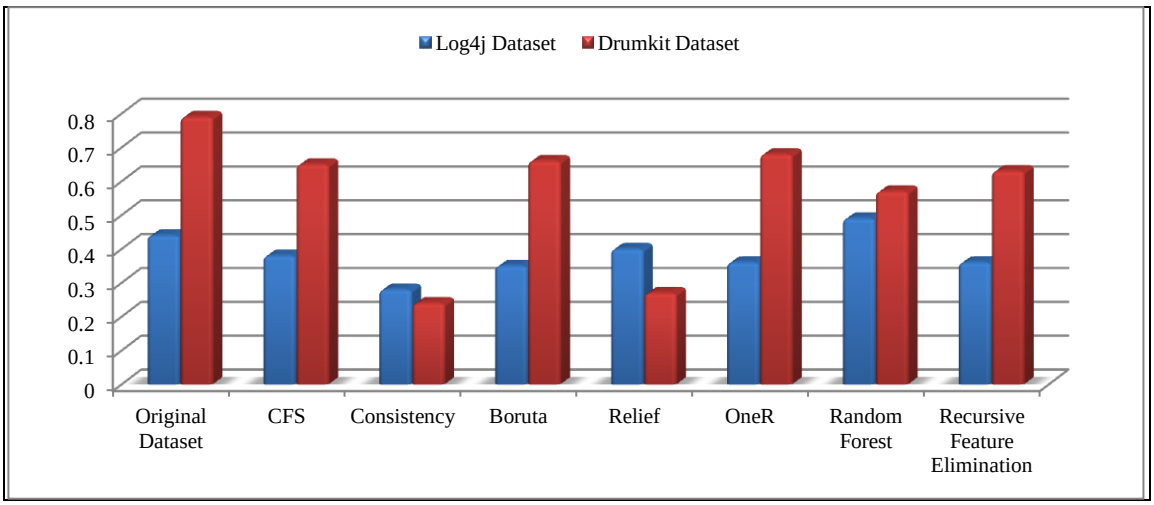

Fig. 5.11 RMSE of Various FS Algorithms with Decision Tree

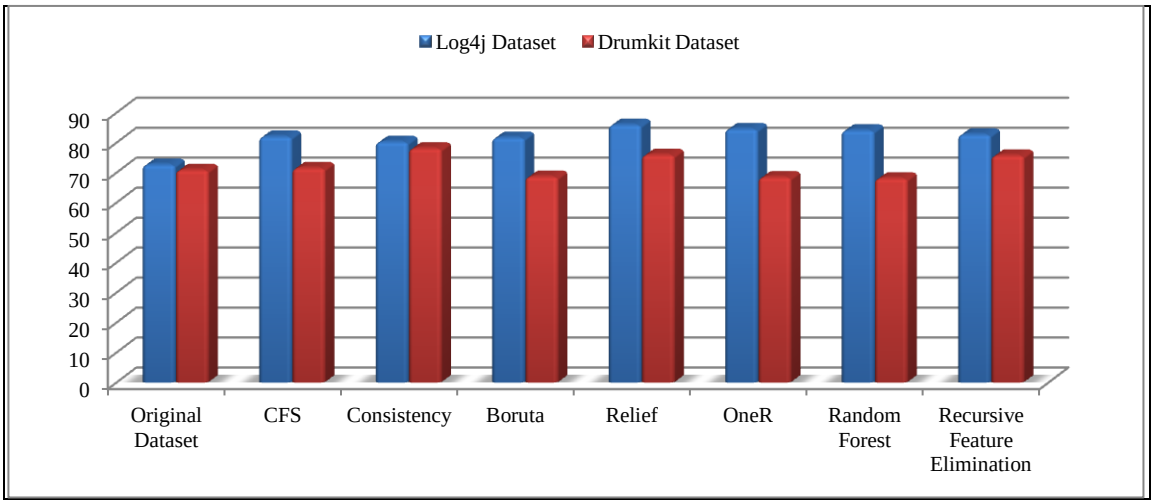

Fig. 5.12 Accuracy of Various FS Algorithms with Decision Tree

TABLE 5.6 Evaluation Results for Cubist with Different FS Algorithms

Cubist								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.50	0.17	0.57	86.15	0.21	0.19	0.52	71.92
CFS	0.58	0.11	0.56	87.89	0.29	0.18	0.50	73.66
Consistency	0.50	0.12	0.57	86.15	0.33	0.18	0.49	71.23
Boruta	0.54	0.12	0.56	86.61	0.24	0.18	0.51	78.68
Relief	0.58	0.12	0.57	86.00	0.25	0.18	0.51	71.92
OneR	0.61	0.12	0.55	88.39	0.38	0.17	0.48	77.62
Random Forest	0.45	0.12	0.58	85.08	0.23	0.18	0.52	71.92
Recursive Feature Elimination	0.51	0.12	0.57	85.08	0.23	0.18	0.52	79.90

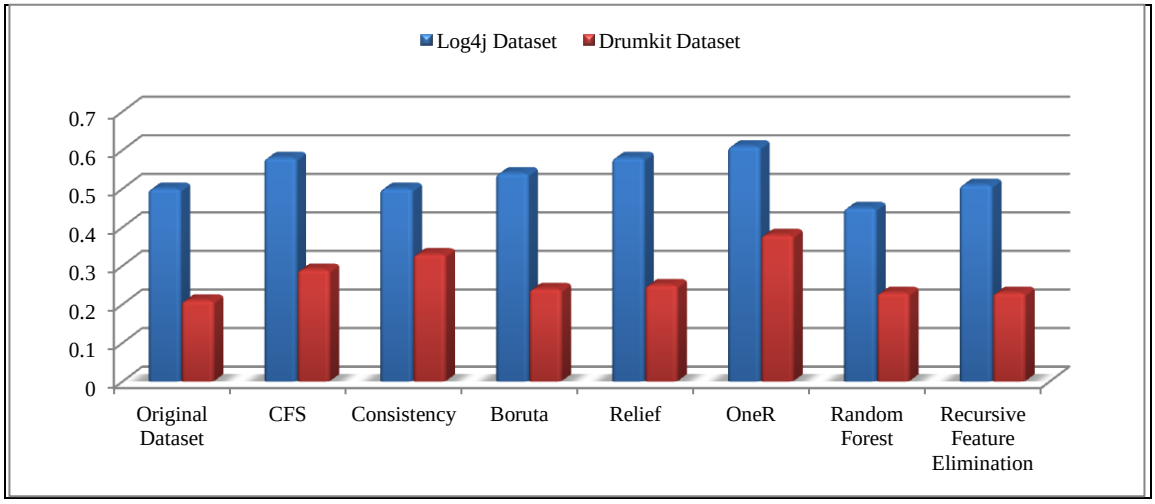

Fig. 5.13 Coefficient of Correlation of Various FS Algorithms with Cubist

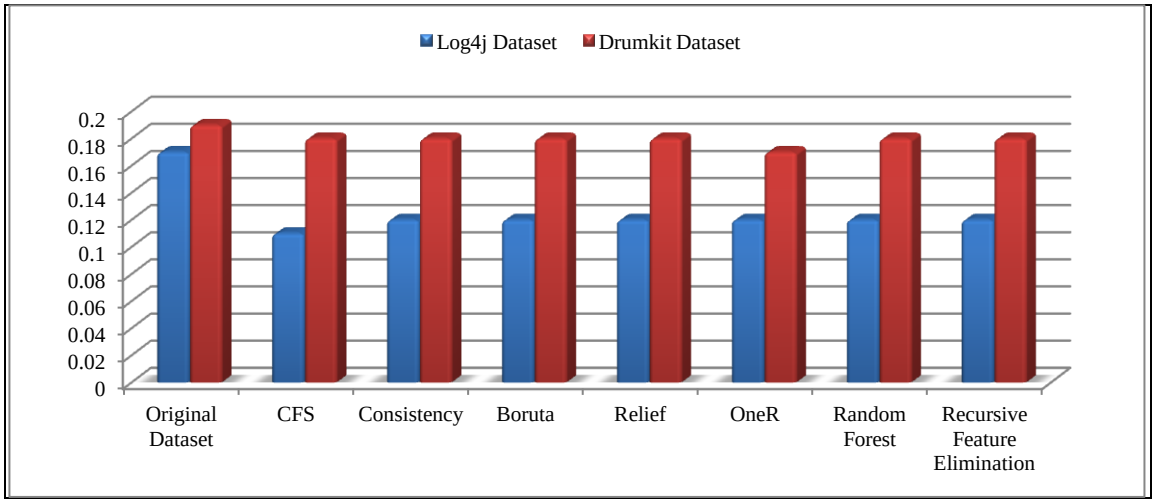

Fig. 5.14 MAE of Various FS Algorithms with Cubist

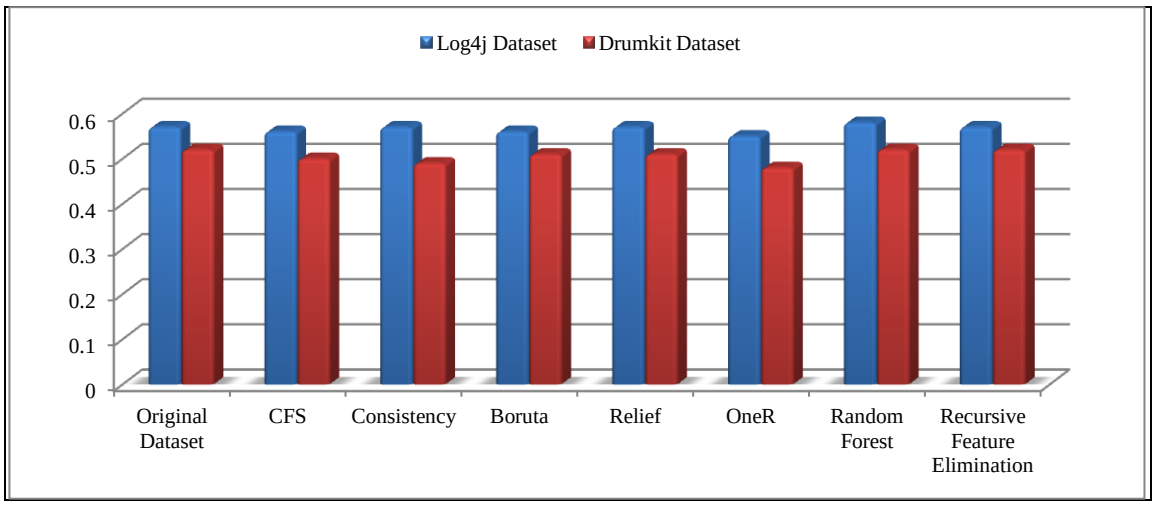

Fig. 5.15 RMSE of Various FS Algorithms with Cubist

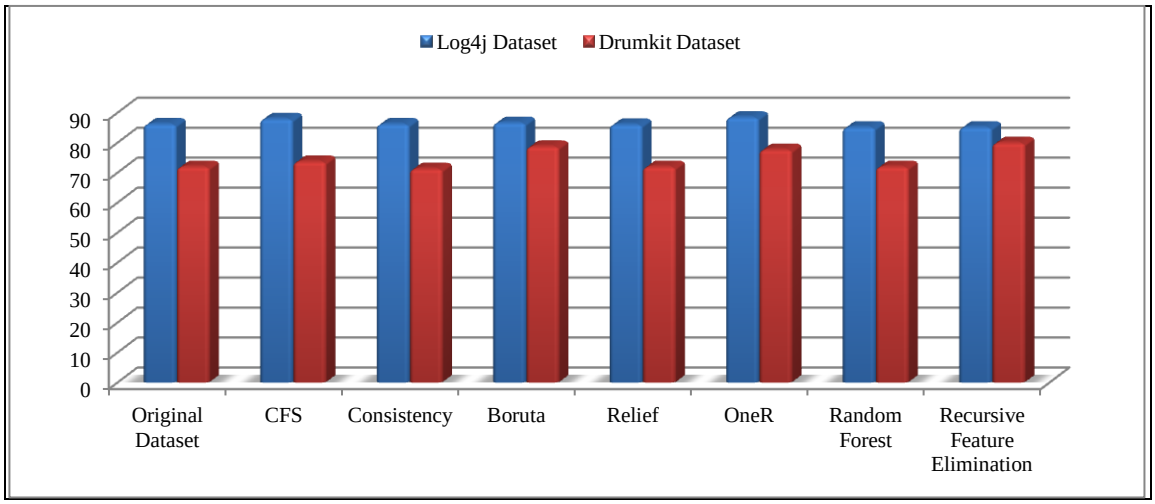

Fig. 5.16 Accuracy of Various FS Algorithms with Cubist

TABLE 5.7 Evaluation Results for Ridge Regression with Variable Selection with Different FS Algorithms

Ridge Regression with Variable Selection								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.50	0.16	0.55	66.75	0.27	0.19	0.40	70.02
CFS	0.56	0.16	0.41	69.71	0.35	0.25	0.44	68.69
Consistency	0.63	0.13	0.31	75.36	0.32	0.17	0.38	71.22
Boruta	0.48	0.15	0.33	66.51	0.36	0.23	0.44	68.80
Relief	0.52	0.13	0.36	69.86	0.42	0.15	0.31	75.14
OneR	0.56	0.13	0.28	69.10	0.35	0.16	0.35	74.52
Random Forest	0.53	0.15	0.47	72.91	0.35	0.25	0.45	67.74
Recursive Feature Elimination	0.51	0.14	0.38	65.14	0.36	0.18	0.38	72.66

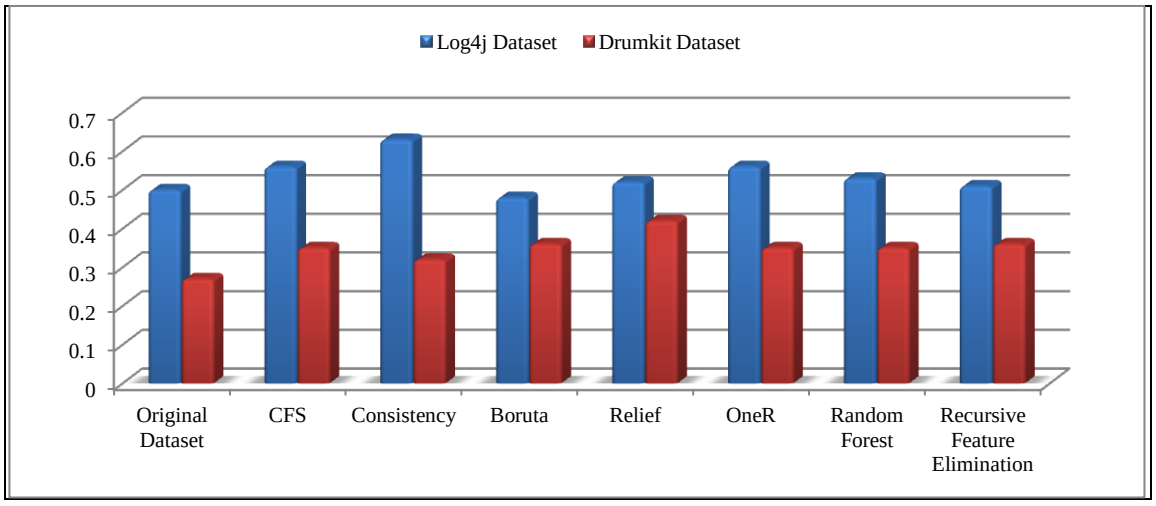

Fig. 5.17 Coefficient of Correlation of Various FS Algorithms with Ridge Regression with Variable Selection

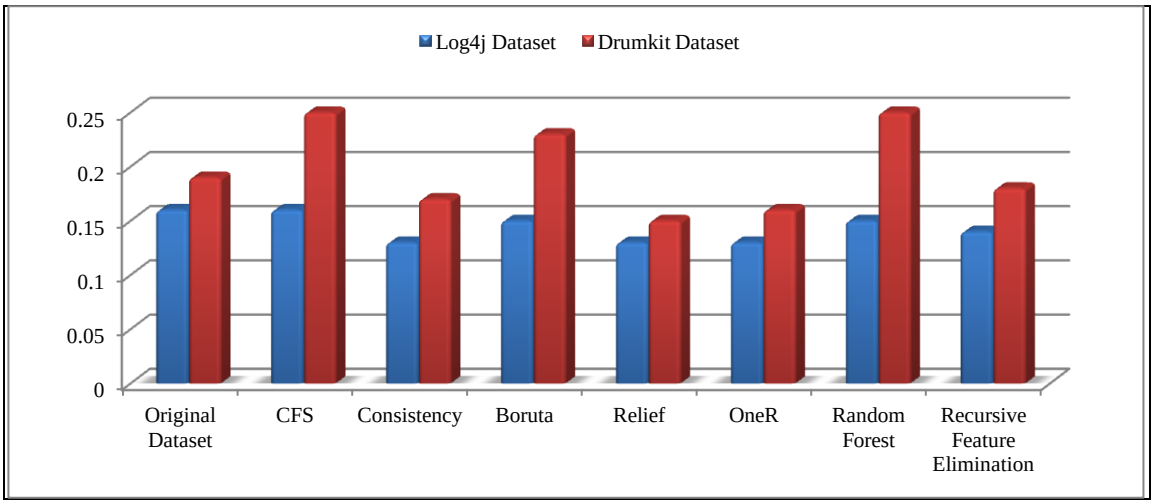

Fig. 5.18 MAE of Various FS Algorithms with Ridge Regression with Variable Selection

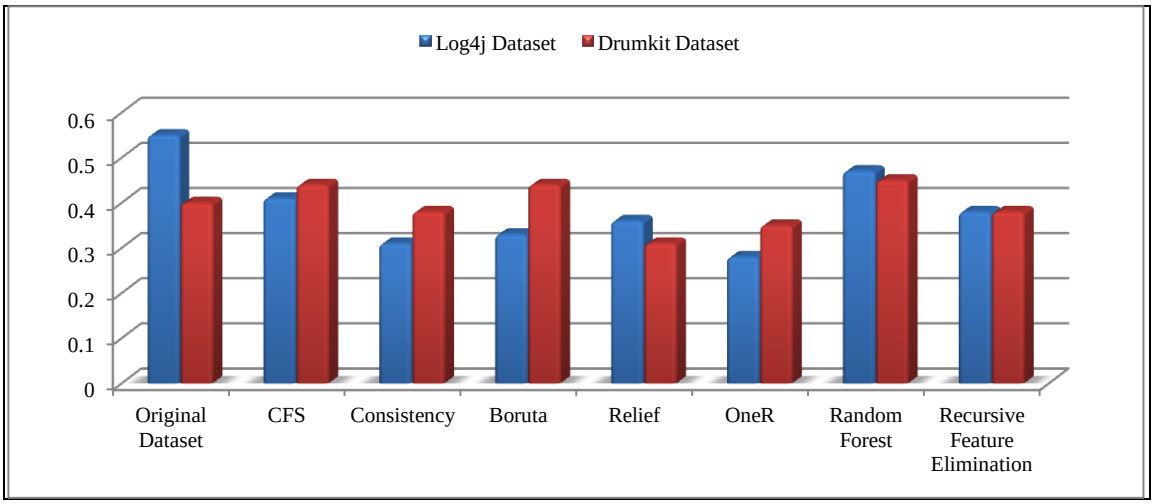

Fig. 5.19 RMSE of Various FS Algorithms with Ridge Regression with Variable Selection

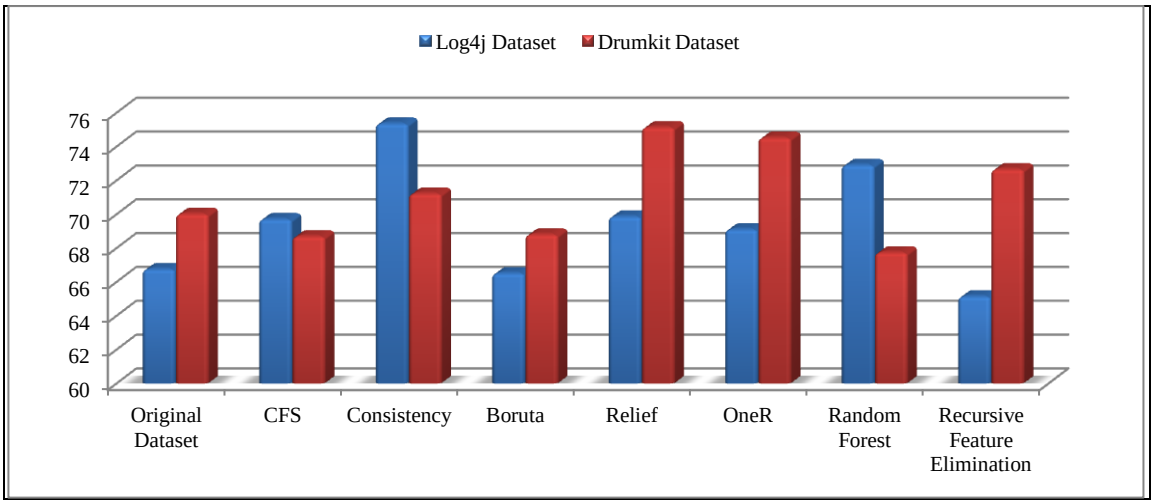

Fig. 5.20 Accuracy of Various FS Algorithms with Ridge Regression with Variable Selection

TABLE 5.8 Evaluation Results for LASSO with Different FS Algorithms

LASSO								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.52	0.15	0.48	68.65	0.29	0.18	0.58	67.42
CFS	0.48	0.13	0.35	71.08	0.4	0.28	0.65	65.85
Consistency	0.61	0.12	0.31	76.23	0.44	0.12	0.41	75.48
Boruta	0.58	0.14	0.4	70.47	0.40	0.25	0.51	67.07
Relief	0.57	0.13	0.46	71.39	0.36	0.14	0.54	71.58
OneR	0.43	0.13	0.4	69.58	0.43	0.13	0.46	70.48
Random Forest	0.43	0.14	0.31	70.93	0.36	0.25	0.52	69.19
Recursive Feature Elimination	0.36	0.15	0.30	75.43	0.40	0.25	0.51	67.07

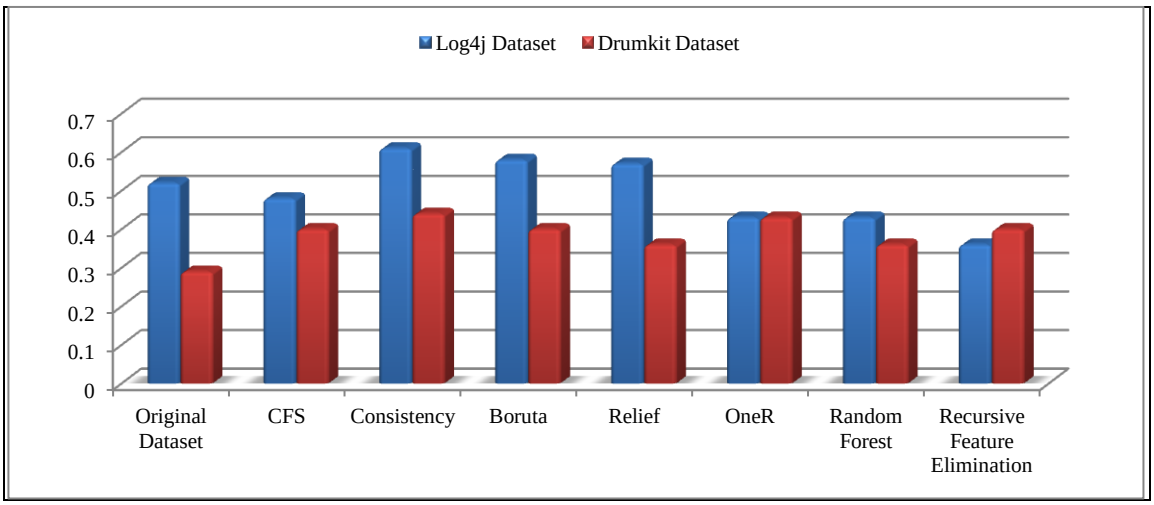

Fig. 5.21 Coefficient of Correlation of Various FS Algorithms with LASSO

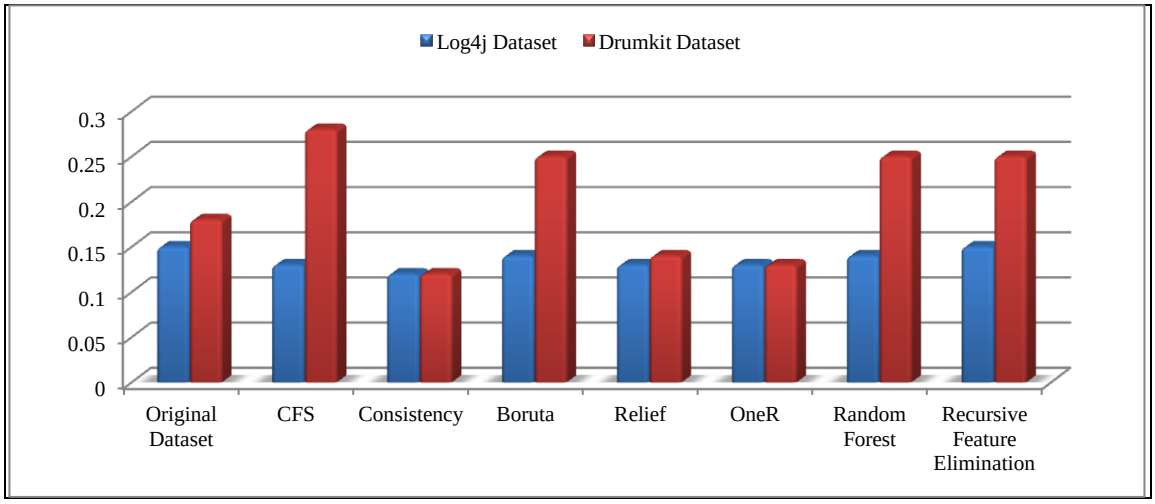

Fig. 5.22 MAE of Various FS Algorithms with LASSO

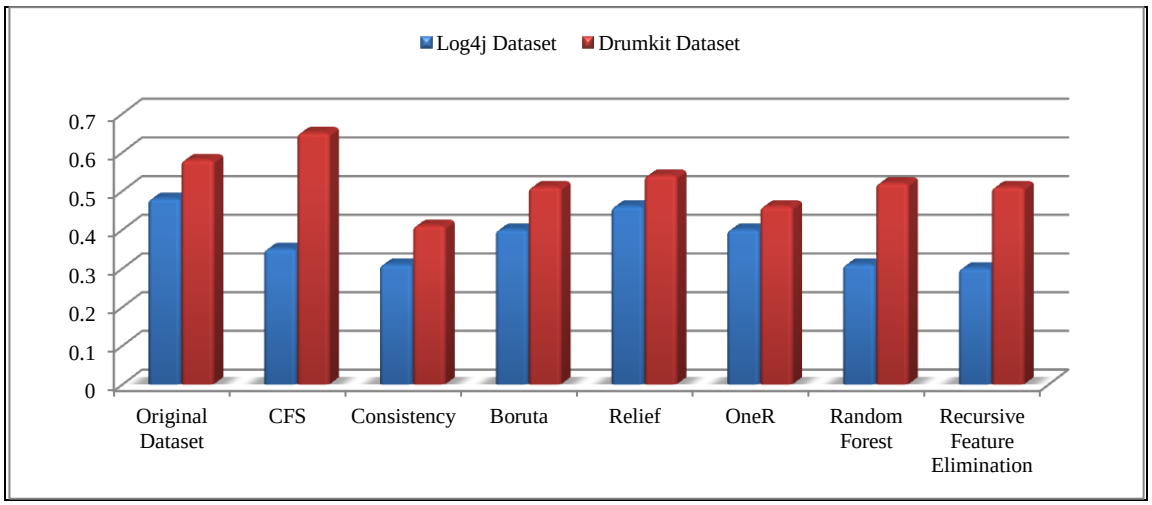

Fig. 5.23 RMSE of Various FS Algorithms with LASSO

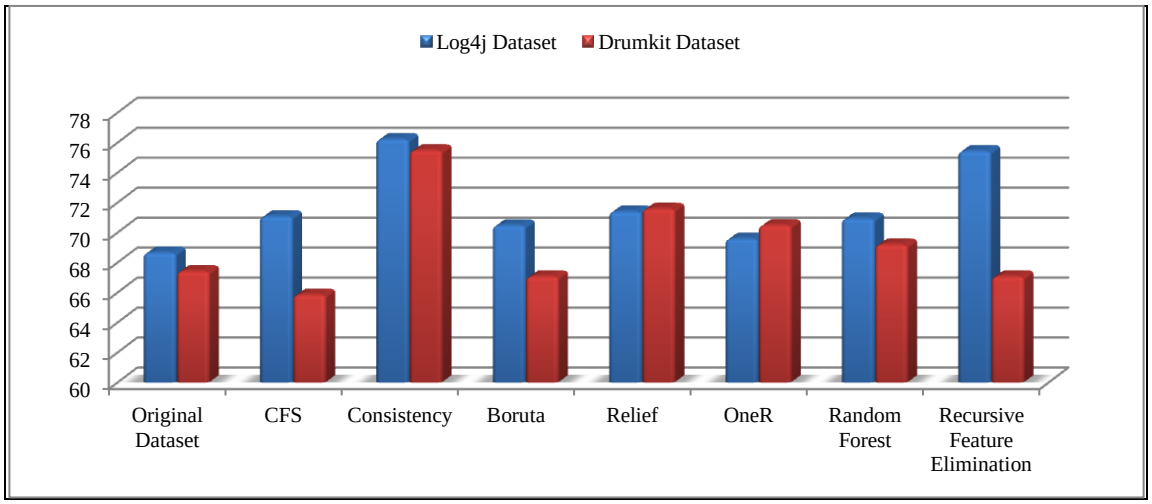

Fig. 5.24 Accuracy of Various FS Algorithms with LASSO

TABLE 5.9 Evaluation Results for Elastic Net with Different FS Algorithms

Elastic Net								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.54	0.17	0.59	61.64	0.32	0.29	0.56	70.88
CFS	0.47	0.13	0.26	85.08	0.28	0.24	0.42	71.40
Consistency	0.57	0.14	0.40	71.54	0.36	0.12	0.25	73.02
Boruta	0.59	0.13	0.28	83.71	0.35	0.25	0.46	69.55
Relief	0.56	0.15	0.49	71.54	0.36	0.13	0.31	73.31
OneR	0.55	0.14	0.36	69.71	0.39	0.14	0.29	74.18
Random Forest	0.53	0.14	0.36	82.34	0.33	0.27	0.52	71.11
Recursive Feature Elimination	0.50	0.12	0.25	84.93	0.35	0.25	0.47	69.55

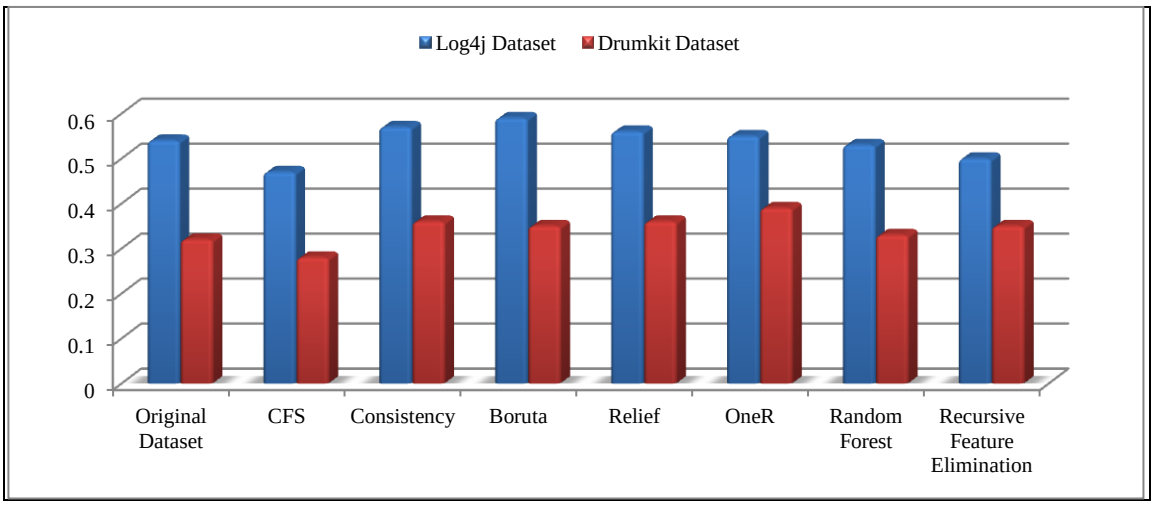

Fig. 5.25 Coefficient of Correlation of Various FS Algorithms with Elastic Net

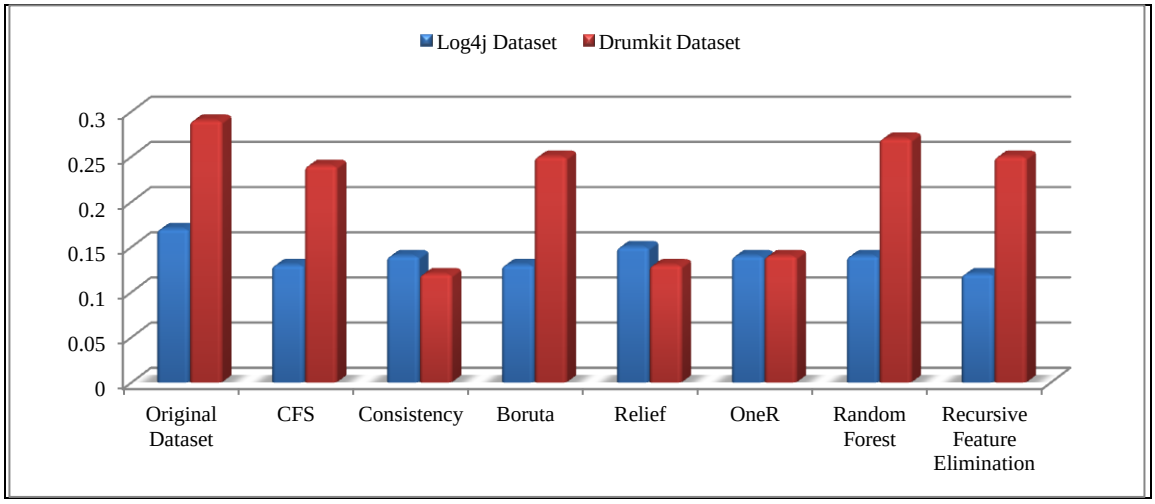

Fig. 5.26 MAE of Various FS Algorithms with Elastic Net

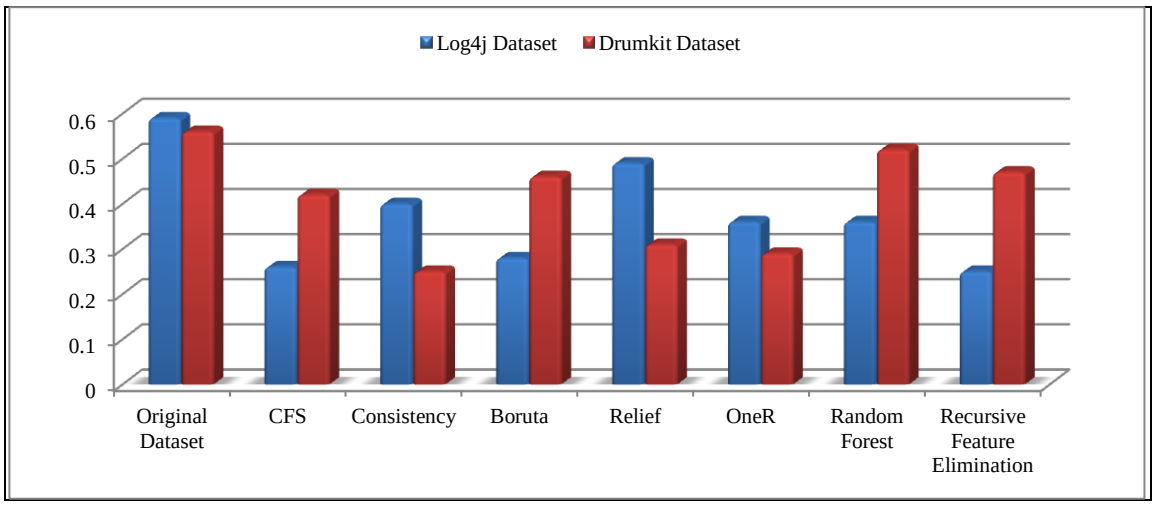

Fig. 5.27 RMSE of Various FS Algorithms with Elastic Net

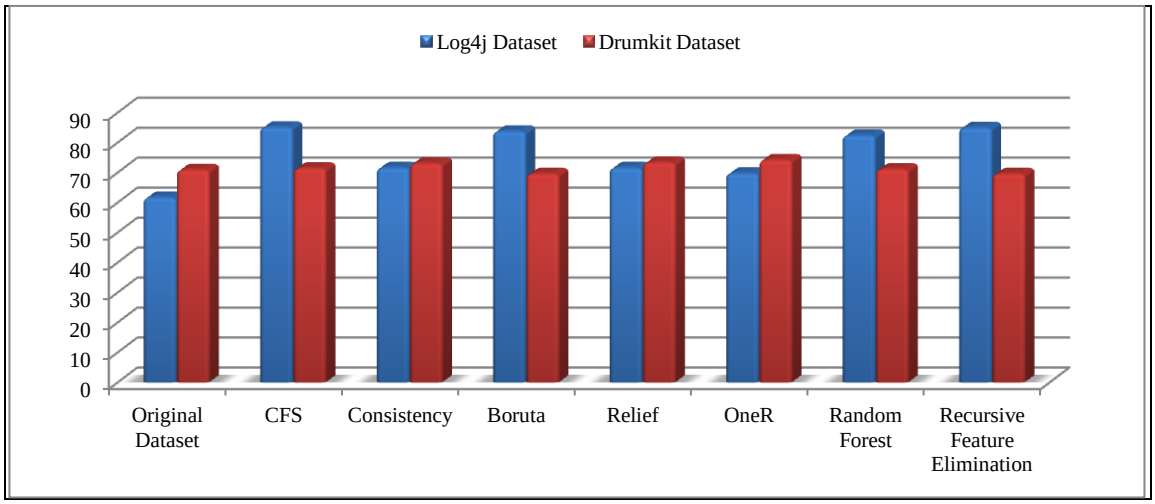

Fig. 5.28 Accuracy of Various FS Algorithms with Elastic Net

TABLE 5.10 Evaluation Results for Random Forest with Different FS Algorithms

Random Forest								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.55	0.25	0.59	72.45	0.33	0.27	0.62	73.08
CFS	0.53	0.16	0.55	81.74	0.33	0.25	0.51	74.47
Consistency	0.54	0.15	0.55	72.60	0.34	0.16	0.51	73.08
Boruta	0.55	0.15	0.55	84.93	0.35	0.25	0.50	72.56
Relief	0.56	0.15	0.52	85.34	0.38	0.16	0.50	73.26
OneR	0.58	0.15	0.55	86.58	0.36	0.14	0.49	79.90
Random Forest	0.53	0.17	0.55	83.87	0.36	0.26	0.51	74.30
Recursive Feature Elimination	0.55	0.15	0.55	83.11	0.36	0.23	0.44	78.80

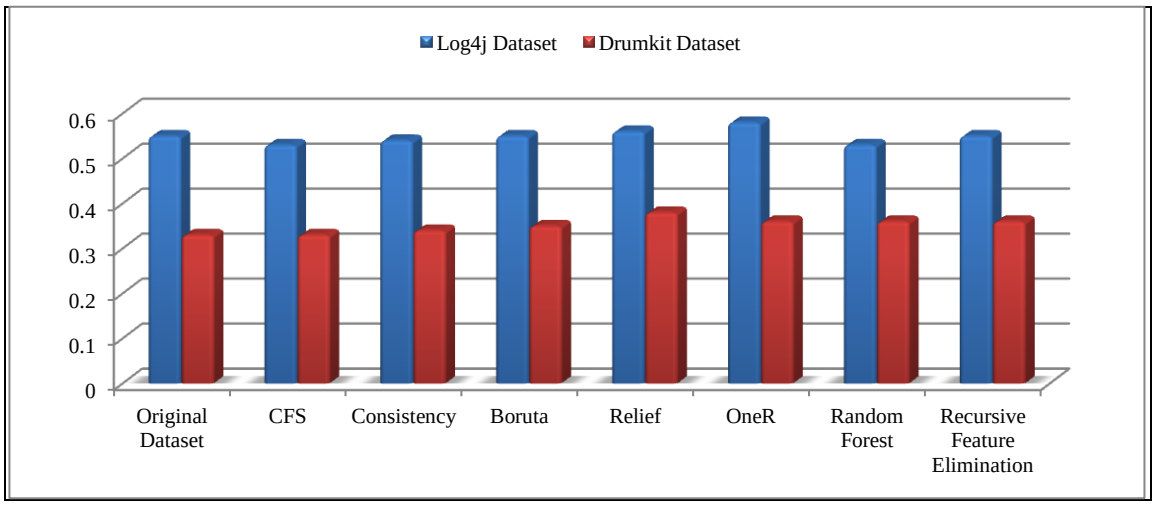

Fig. 5.29 Coefficient of Correlation of Various FS Algorithms with Random Forest

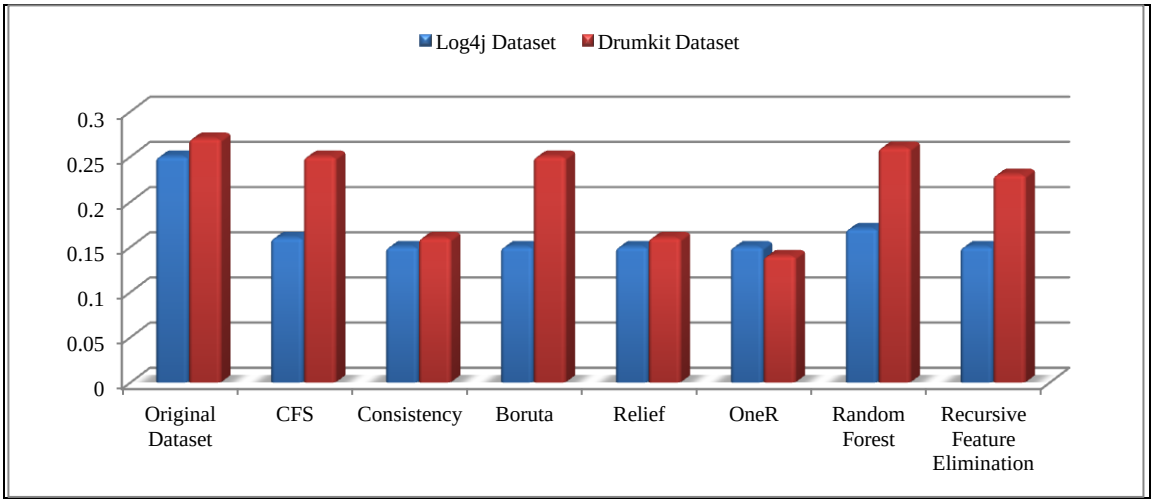

Fig. 5.30 MAE of Various FS Algorithms with Random Forest

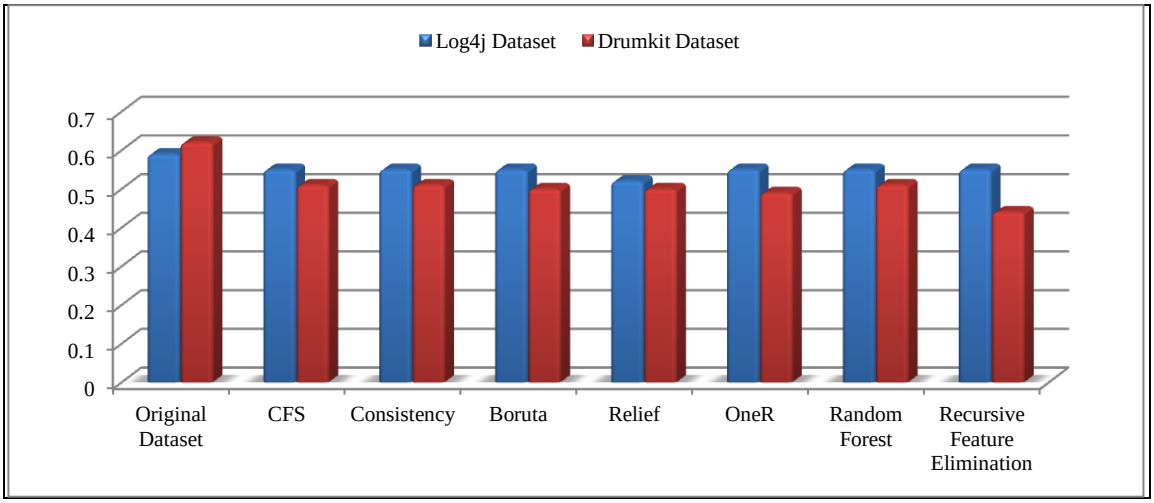

Fig. 5.31 RMSE of Various FS Algorithms with Random Forest

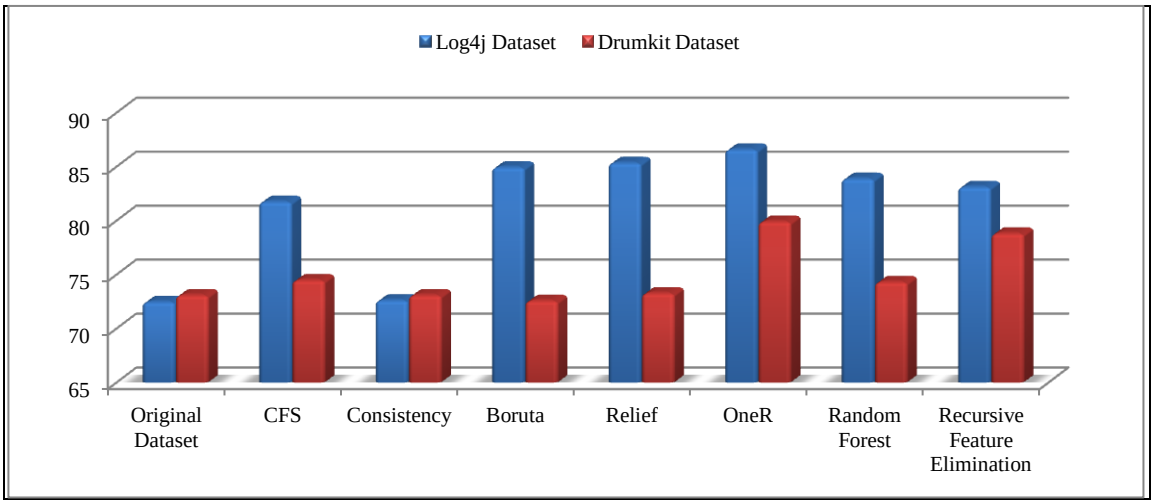

Fig. 5.32 Accuracy of Various FS Algorithms with Random Forest

TABLE 5.11 Evaluation Results for Principal Component Analysis with Different FS Algorithms

Principal Component Analysis								
FS Algorithm	**Apache Log4j**				**Drumkit**			
	r	**MAE**	**RMSE**	**Accuracy**	**r**	**MAE**	**RMSE**	**Accuracy**
Original Dataset	0.27	0.14	0.47	73.06	0.32	0.27	0.52	71.23
CFS	0.41	0.12	0.28	72.60	0.28	0.26	0.56	68.33
Consistency	0.54	0.13	0.31	76.71	0.38	0.14	0.30	74.35
Boruta	0.45	0.13	0.30	74.23	0.30	0.26	0.55	70.19
Relief	0.43	0.12	0.34	70.34	0.42	0.13	0.28	74.70
OneR	0.45	0.14	0.34	72.04	0.43	0.13	0.29	75.39
Random Forest	0.49	0.12	0.31	71.71	0.40	0.25	0.51	71.53
Recursive Feature Elimination	0.50	0.13	0.28	73.32	0.30	0.26	0.55	70.19

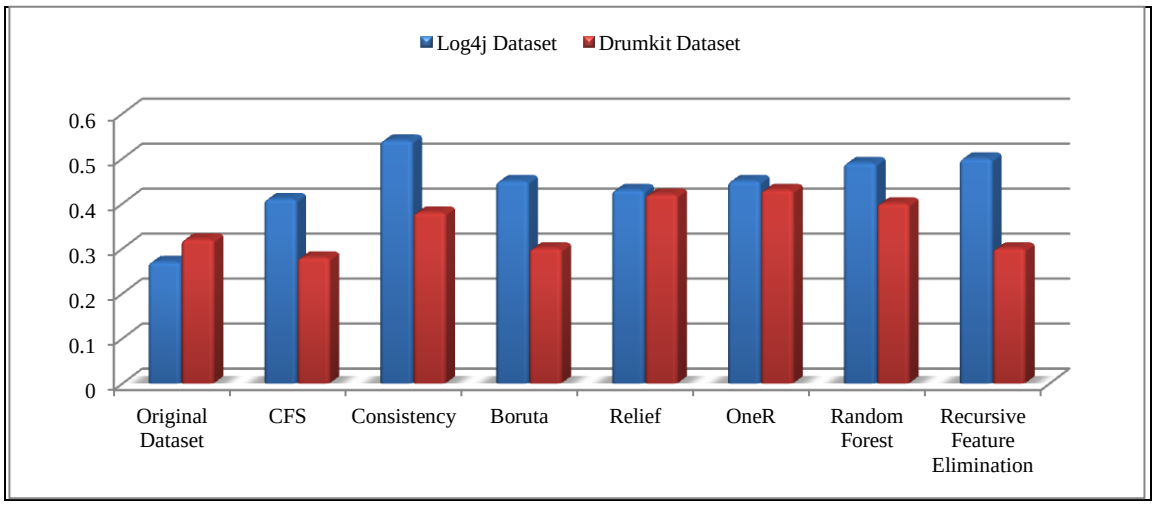

Fig. 5.33 Coefficient of Correlation of Various FS Algorithms with Principal Component Analysis

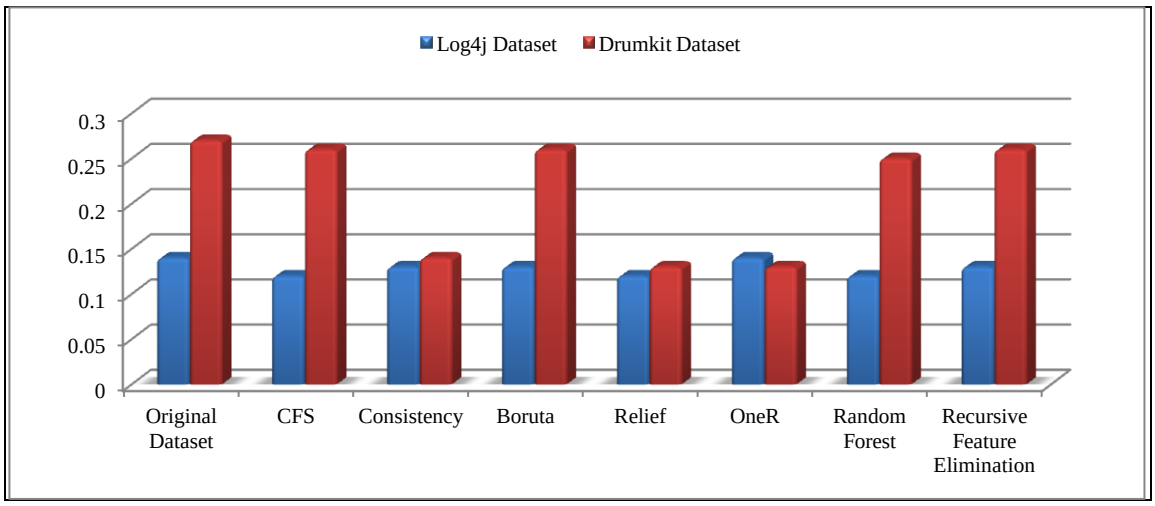

Fig. 5.34 MAE of Various FS Algorithms with Principal Component Analysis

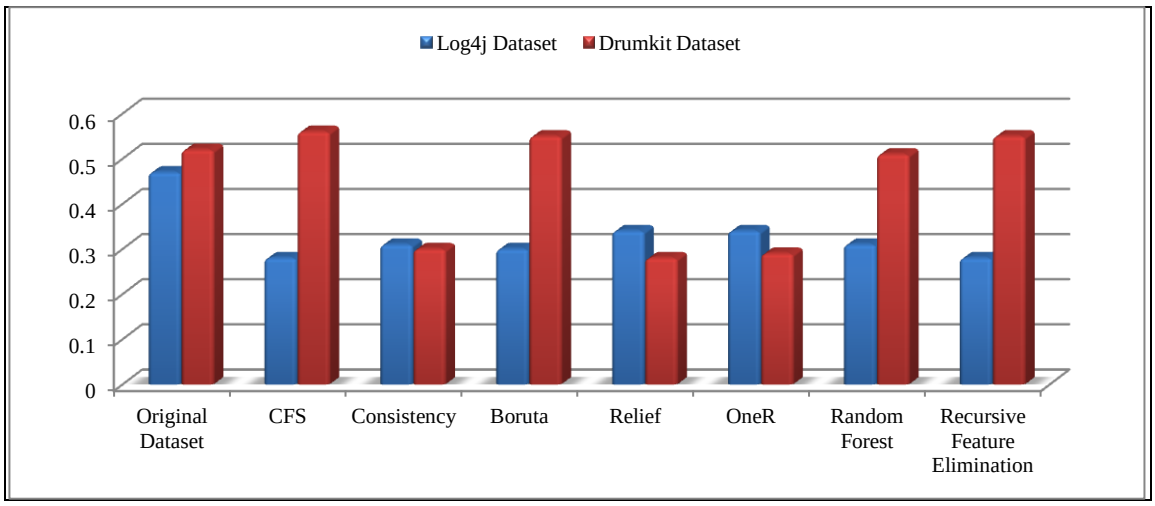

Fig. 5.35 RMSE of Various FS Algorithms with Principal Component Analysis

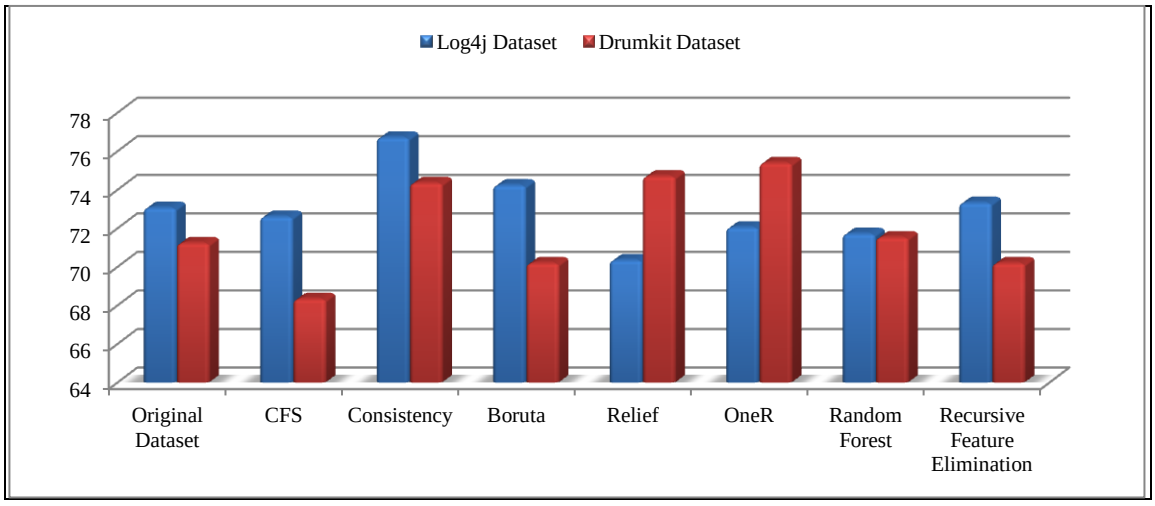

Fig. 5.36 Accuracy of Various FS Algorithms with Principal Component Analysis

5.2 Comparison with Existing Work

The results of different combinations of FS and Linear regression, General Regression Neural Network and Decision Tree have been compared with exiting results [27] as shown in TABLE 5.12 and 5.13 for Apache Log4j and Drumkit respectively. It is evident from the results that ML (Linear Regression, General Regression Neural Network and Decision Tree) algorithms give better results using FS.

- Fig. 5.37 and Fig. 5.40 compare the values of MAE of the existing work by Chug and Malhotra [27] and present work (using FS) for Apache Log4j and Drumkit datasets respectively. The graphs show that there is decrease in MAE value of prediction models wherein only relevant features as input.
- Similarly Fig. 5.38 and Fig. 5.41 display the old and new RMSE values for both the datasets. Some ML algorithms like shows drastic change in RMSE value when FS is applied.
- Fig. 5.39 and Fig. 5.42 depict the Accuracy of different prediction models on two open source datasets: Apache Log4j and Drumkit. It shows the different values of accuracy calculated with and without feature selection techniques.
- The value of MAE has fallen from 0.57 to 0.34 and 0.36 to 0.14 [27] when Linear Regression is used with Relief FS in Apache Log4j and Drumkit respectively as compared to existing work. Thus, it can be established that Linear Regression works good with Relief FS.
- Further, General Regression Neural Network shows improved results with CFS FS technique for both the datasets as compared to Chug and Malhotra's existing results [27].
- Decision tree with Recursive Feature Elimination for Apache Log4j and with Consistency with Drumkit shows fall in error (MAE and RMSE) and increase in Accuracy levels when compared with old results [27].

TABLE 5.12 Performance Comparison of FS and ML on Apache Log4j Dataset

FS	ML Model	MAE	RMSE	Accuracy
Linear Regression	**Existing Results**	0.36	0.49	70.00
	CFS	0.26	0.35	71.5
	Consistency	0.20	0.32	75.23
	Boruta	0.38	0.47	71.01
	Relief	0.18	0.34	78.65
	OneR	0.26	0.44	69.5
	Random Forest	0.38	0.46	71.06
	Recursive Feature Elimination	0.30	0.46	70.05
General Regression Neural Network	**Existing Results**	0.32	0.32	80.00
	CFS	0.26	0.38	82.60
	Consistency	0.34	0.40	72.60
	Boruta	0.30	0.44	73.11
	Relief	0.29	0.38	78.99
	OneR	0.32	0.52	71.13
	Random Forest	0.30	0.47	74.79
	Recursive Feature Elimination	0.30	0.28	74.33
Decision Tree	**Existing Results**	0.42	0.43	71.00
	CFS	0.28	0.38	82.00
	Consistency	0.25	0.28	80.23
	Boruta	0.26	0.35	81.64
	Relief	0.14	0.40	85.93
	OneR	0.26	0.36	84.53
	Random Forest	0.26	0.49	84.07
	Recursive Feature Elimination	0.24	0.36	82.70

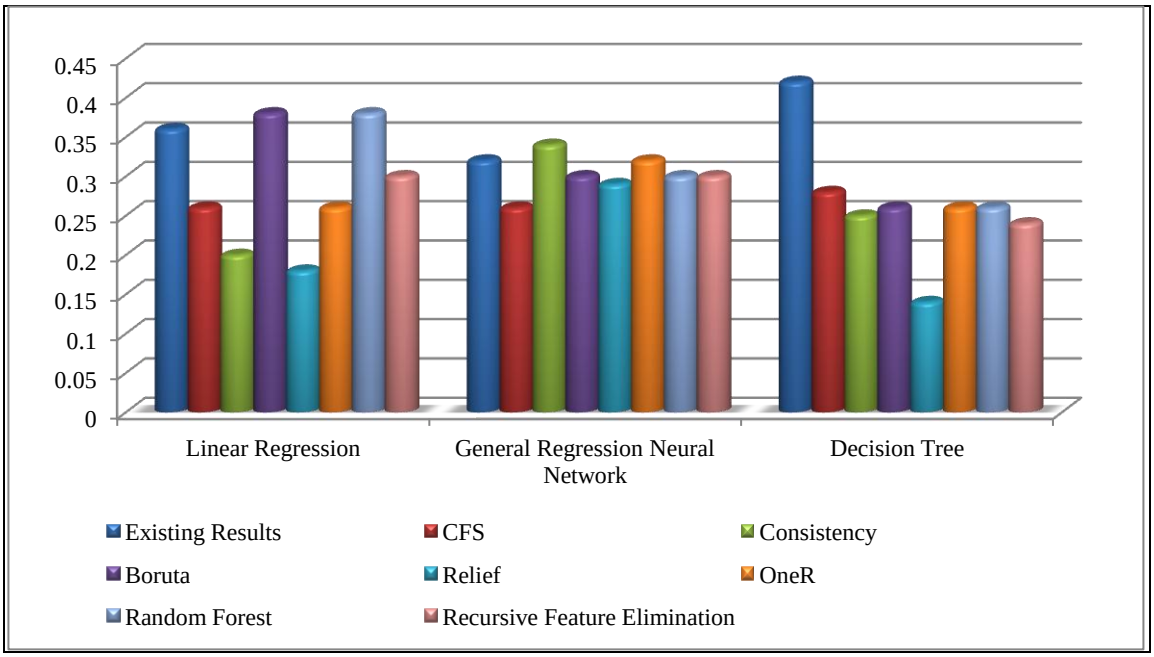

Fig. 5.37 Comparison of Existing and New MAE Values of Apache Log4j Dataset

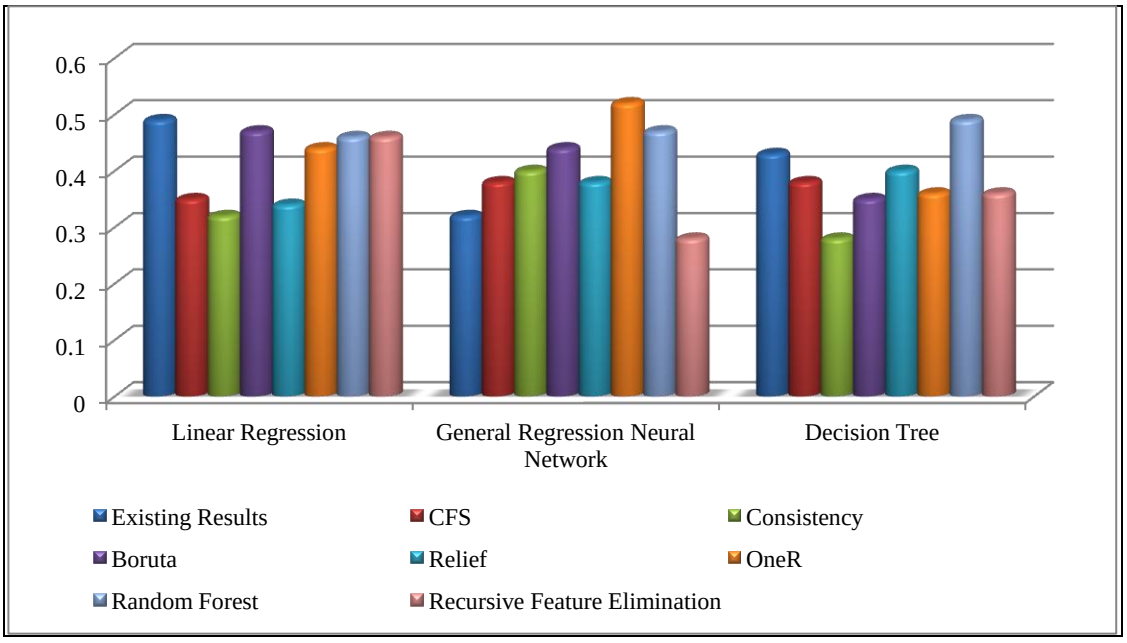

Fig. 5.38 Comparison of Existing and New RMSE Values of Apache Log4j Dataset

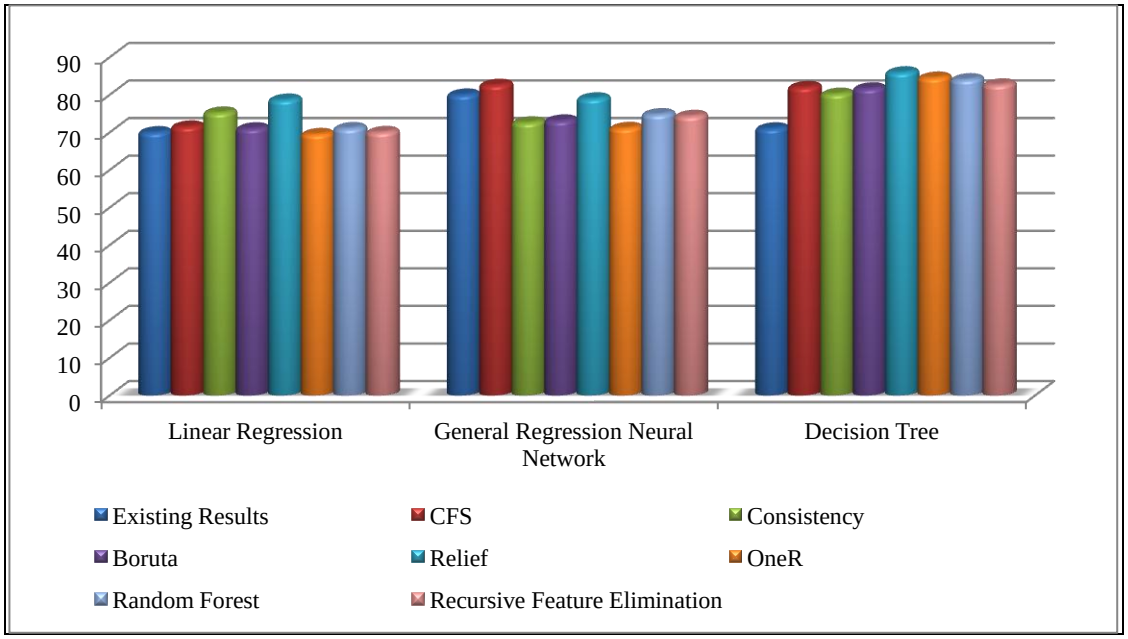

Fig. 5.39 Comparison of Existing and New Accuracy Values of Apache Log4j Dataset

TABLE 5.13 Performance Comparison of FS and ML on Drumkit Dataset

FS	ML Model	MAE	RMSE	Accuracy
Linear Regression	**Existing Results**	0.57	0.66	78.00
	CFS	0.48	0.53	78.50
	Consistency	0.38	0.23	82.68
	Boruta	0.50	0.65	81.11
	Relief	0.34	0.35	85.73
	OneR	0.50	0.59	79.17
	Random Forest	0.50	0.58	82.06
	Recursive Feature Elimination	0.54	0.63	80.50
General Regression Neural Network	**Existing Results**	0.38	0.38	81.00
	CFS	0.19	0.60	85.99
	Consistency	0.26	0.67	80.15
	Boruta	0.37	0.33	81.01

	Relief	0.28	0.65	79.94
	OneR	0.35	0.40	79.13
	Random Forest	0.54	0.60	74.79
	Recursive Feature Elimination	0.60	0.85	74.33
Decision Tree	**Existing Results**	0.58	0.86	67.00
	CFS	0.53	0.65	71.50
	Consistency	0.12	0.24	78.08
	Boruta	0.52	0.66	68.64
	Relief	0.14	0.27	75.91
	OneR	0.52	0.68	68.44
	Random Forest	0.52	0.57	68.07
	Recursive Feature Elimination	0.48	0.63	75.70

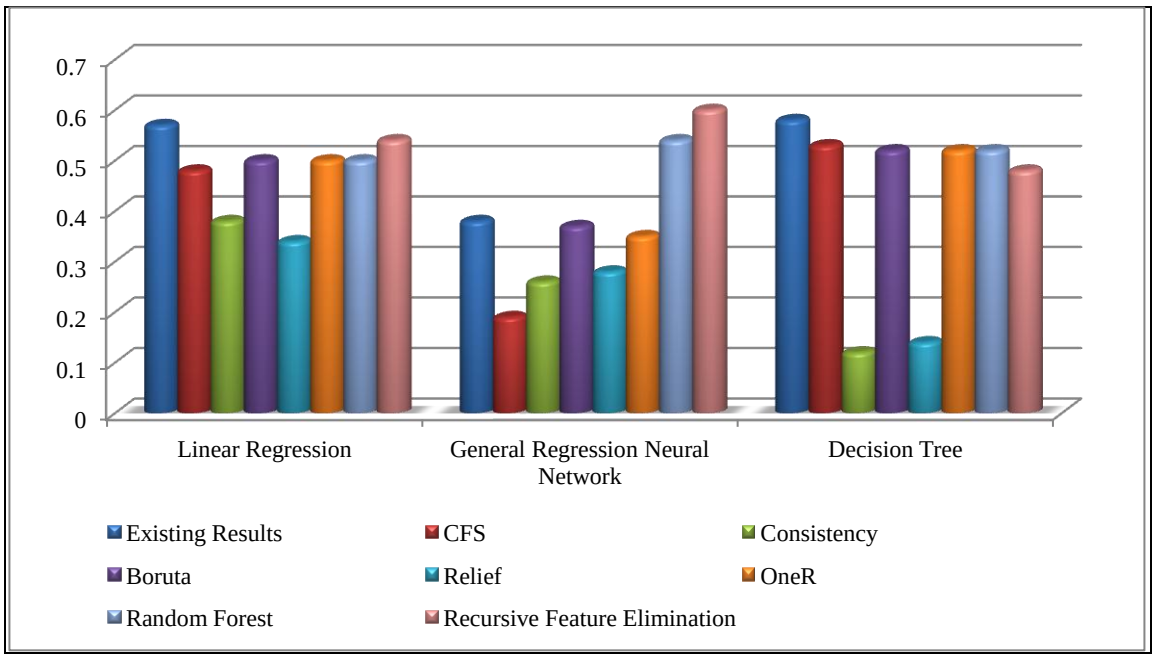

Fig. 5.40 Comparison of Existing and New MAE Values of Drumkit Dataset

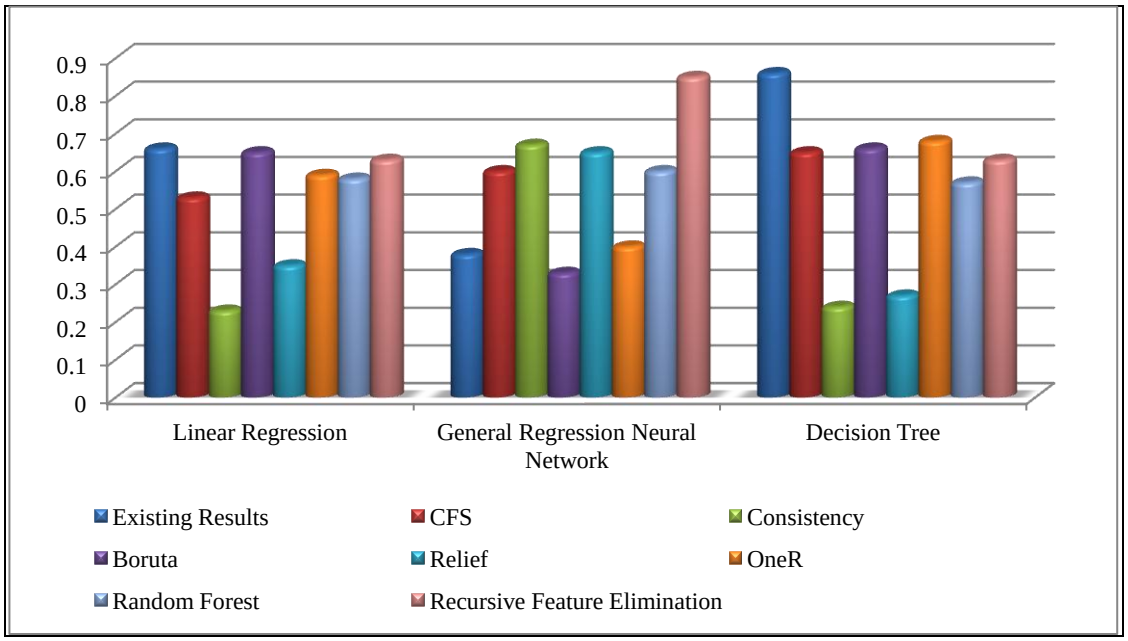

Fig. 5.41 Comparison of Existing and New RMSE Values of Drumkit Dataset

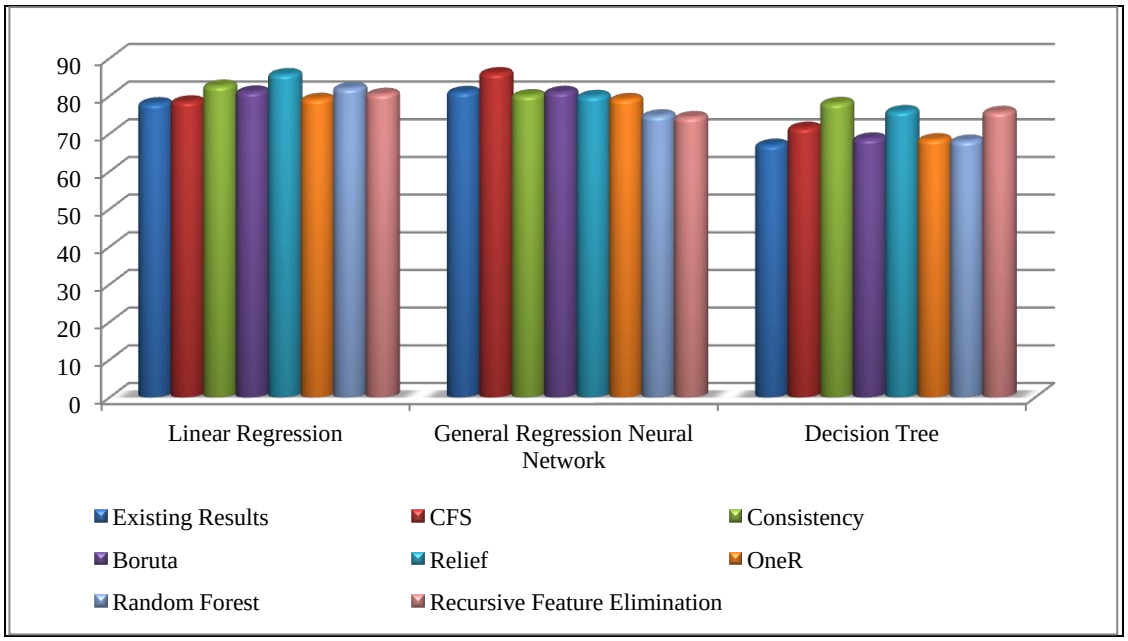

Fig. 5.42 Comparison of Existing and New Accuracy Values of Drumkit Dataset

CHAPTER – 6 CONCLUSION & FUTURE SCOPE

6.1 Conclusion

In this empirical study, the performance and error level of seven FS algorithm – CFS, Consistency, Boruta, Relief, OneR, Random Forest and Recursive Feature Elimination; with nine ML algorithms i.e. Linear Regression, General Regression Neural Network, Decision Tree, Cubist, Ridge Regression with Variable Selection, LASSO, Elastic Net, Random Forest and Principal Component Analysis for prediction of CHANGE in two versions of two open source software was evaluated. Datasets of open source software systems Apache Log4j and Drumkit were considered for this study. As it has been observed, FS techniques help in reducing the error in CHANGE prediction and provide better results. Further, for large projects, effort, time and cost required to analyze a smaller set of metrics is apparently less as compared to analyzing all the metrics. According to the results evaluated in this study, the following inferences have been made:

- LCO, LOC, SLOC-P, SLOC-L, RFC and Ce have been observed to be the most significant metrics for predicting structural quality of open source software.
- The combination of Linear Regression with Relief FS algorithm gave improved results for both the open source software (Apache Log4j and Drumkit) selected in this study.
- General Regression Neural Network ML algorithm gives comparatively better results after feature reduction using CFS FS method on both Apache Log4j and Drumkit dataset.
- In case of Decision Tree, FS technique ‘Recursive Feature Elimination’ gave enhanced prediction evaluation results for Apache Log4j and FS technique ‘Consistency’ for Drumkit dataset.
- Cubist ML algorithm works best when it is applied after OneR FS method for both the open source software systems selected in this study.

- Performance of Ridge Regression with Variable Selection was found to be superior with Consistency FS algorithm as compared to with original dataset without any feature cutback for Apache Log4j and with Relief FS algorithm for Drumkit dataset.
- LASSO gave superior results with Consistency FS method for both open source software datasets.
- Elastic Net gave best results with Recursive Feature Elimination FS algorithm for Apache Log4j and with Consistency FS algorithm for Drumkit.
- For Apache Log4j Random Forest with OneR FS method gave improved results as compared to with original dataset without feature reduction. Similarly, for Drumkit dataset also Random Forest with OneR gave better results.
- The combination of Consistency FS technique with Principal Component Analysis ML algorithm for Apache Log4j and OneR FS technique with Principal Component Analysis for Drumkit show superior results in comparison to prediction results of original dataset without any FS.
- It was analyzed that for Random Forest algorithm there is not much improvement in terms of accuracy prediction parameters when different FS methods are applied, as it is an advance ML algorithm which internally implements FS before the prediction process.
- It was observed from the evaluation results and graphs plotted for r, MAE, RMSE and Accuracy that Consistency FS method gave best prediction results after redundant feature removal for maximum ML algorithms.
- Further, OneR FS algorithm gave enhanced results in most combinations after Consistency FS method.

This study confirms that applying FS algorithms before ML prediction models helps in reduction of irrelevant and redundant features, thus, saving time, cost and effort and giving more precise evaluation results. The final conclusions on best combinations of FS and ML algorithms are summarized in TABLE 6.1.

The only drawback perceived in this study was that the results cannot be replicated as for generalization of the results the prediction models were trained repeatedly with random sets of training and testing data.

TABLE 6.1 Final Conclusions on Best FS and ML Combinations

S.No.	ML Model	Best FS Technique for Apache Log4j Dataset	Best FS Technique for Drumkit Dataset
1.	Linear Regression	Relief	Relief
2.	General Regression Neural Network	CFS	CFS
3.	Decision Tree	Recursive Feature Elimination	Consistency
4.	Cubist	OneR	OneR
5.	Ridge Regression with Variable Selection	Consistency	Relief
6.	LASSO	Consistency	Consistency
7.	Elastic Net	Recursive Feature Elimination	Consistency
8.	Random Forest	OneR	OneR
9.	Principal Component Analysis	Consistency	OneR

6.2 Future Scope

In this empirical study, we have originally considered twenty-nine object-oriented metrics as input. It has been observed that not all the metrics are important for the software maintenance prediction. FS algorithms have improved the efficiency of the ML algorithms. The major application of this study is found in the software engineering domain to have quantitative measure of the quality of the software based on the object-oriented metrics.

The research can be enhanced on this field by:

- Collecting more metrics (features) using different tools available and then the prediction results can be analyzed.
- Domain specific software can be taken for comparative analysis.
- Different FS algorithms can be used and they can be tested with different ML algorithms so as to find the best combinations of FS algorithms and ML.
- The ML models work much better if we have larger data set so that the training can be unbiased. Larger dataset can be collected and analyzed for more effective models. One of the ways to achieve this is aggregating different domain specific software's data and then analyzing them.
- A comparative study of evolutionary algorithms and traditional algorithms can also be done.

REFERENCES

[1] R. Malhotra and A. Chug, *"Software Maintainability: Systematic Literature Review and Current Trends"* in International Journal of Software Engineering and Knowledge Engineering, vol. 26.8, pp. 1221-1253, 2016.

[2] G. Chandrashekar and F. Sahin, *"A survey on feature selection methods"* in Computers and Electrical Engineering, vol. 40, pp. 16-28, 2014.

[3] R. Kohavi and G.H. John, *"Wrappers for feature subset selection"* in Artificial Intelligence, vol. 97.1-2, pp. 273-324, 1997.

[4] A.L Blum, P. Langley, *"Selection of relevant features and examples in machine learning"* in Artificial Intelligence, vol. 97, pp. 245–70, 1997.

[5] I. Guyon and A. Elisseeff, *"An introduction to variable and feature selection"*, Journal of Machine Learning Research, pp. 1157-1182, 2003.

[6] W. Li and S. Henry, *"Object-oriented metrics that predict maintainability"* in Journal of Systems and Software, vol. 23.2, pp. 111-122, 1993.

[7] W. Li, *"Another metric suite for object–oriented programming"* in The Journal of Systems and Software, vol. 44.2, pp. 155-162, 1998.

[8] S. Chidamber and C.F. Kemerer, *"A metrics suite for object-oriented design"* in IEEE Transactions on software engineering, vol. 20.6, pp. 476-493, 1994.

[9] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, *"Empirical study of object-oriented metrics"* in Journal of Object Technology, vol. 5.8, pp. 149-173, 2006.

[10] M. Riaz, E. Mendes and E. Tempero, *"A systematic review of software maintainability prediction and metrics"* in Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2009.

[11] F. Fioravanti and P. Nesi, *"Estimation and prediction metrics for adaptive maintenance effort of object-oriented metrics"* in IEEE Transactions on Software Engineering, vol. 27.12, pp. 1062-1084, 2001.

[12] R. Malhotra and A. Chug, *"Software maintainability prediction using machine learning algorithms"* in Software Engineering: An International Journal (SEIJ), vol. 2.2, 2012.

[13] Y. Zhou and H. Leung, *"Predicting object-oriented software maintainability using multivariate adaptive regression splines"* in Journal of Systems and Software, vol. 80, pp. 1349-1361, 2007.

[14] A.Heiat, *"Comparison of artificial neural network and regression models for estimating software development effort"* in Information and Software Technology, vol. 44, pp. 911-922, 2002.

[15] A. Kaur and K. Kaur, *"Statistical comparison of modelling methods for software maintainability prediction"* in International Journal of Software Engineering and Knowledge Engineering, vol. 23.06, pp. 743-774, 2013.

[16] S.K. Dubey, A. Rana and Y. Dash, *"Maintainability prediction of object-oriented software system by multilayer perceptron model"* in ACM SIGSOFT Software Engineering Notes, vol. 37.5, pp. 1-4, 2012.

[17] M.A. Ahmed and A.A.J. Hamdi, *"Machine learning approaches for predicting mainatinability: a fuzzy-based transparent model"* in IET Software, vol. 7.6, pp. 317-326, 2013.

[18] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, *"Application of artificial neural network for predicting maintainability using object-oriented metrics"* in Transactions on Engineering, Computing and Technology, vol. 15, pp. 285-289, 2006.

[19] C.V. Koten and A.R. Gray, *"An application of bayesian network for predicting object-oriented software maintainability"* in Information and Software Technology, vol. 48.1, pp. 59-67, 2006.

[20] G.H. John, R. Kohavi and K. Pfelgar, *"Irrelevant features and the subset selection problem"* in Machine Learning: Proceedings of the Eleventh International Conference, pp. 121-129, 1994.

[21] M.A. Hall and G. Holmes, *"Benchmarking attribute selection techniques for discrete class data mining"* in IEEE Transactions on Knowledge and Data Engineering, vol. 15.3, 2003.

[22] Z. Chen, T. Menzies, D. Port and D. Boehm, *"Finding the right data for software cost modelling"* in IEEE Software, vol. 22.6, pp. 38-46, 2005.

[23] S. Manchanda and A. Chug, *"CFS based feature subset selection for software maintenance prediction"* in International Journal of Advance Foundation and Research in Computer (IJAFRC), vol. 2.5, 2015.

[24] M. Dash and H. Liu, *"Consistency based search in feature selection"* in Artificial Intelligence, vol. 151, pp. 155-176, 2003.

[25] I. Inza, P. Larranaga, R. Etxeberria and B. Sierra, *"Feature subset selection by bayesian network - based optimization"* in Aritificial Intelligence, vol. 123, pp. 157-184, 2000.

[26] A. Jain and D. Zongker, *"Feature selection: Evaluation, application and small sample performance"* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19.2, 1997.

[27] A. Chug and R. Malhotra, *"Benchmarking framework for maintainability prediction of open source software using object-oriented metrics"* in International Journal of Innovative Computing, Information and Control, vol.12.2, April 2016.

[28] R. Gunnalan, T. Menzies, K. Appukutty and A. Srinivasan, *"Feature subset selection with TAR2less"*, 2003.

[29] S. Khalid, T. Khalil and S. Nasreen, *"A survey of feature selection and feature extraction techniques in machine learning"*, Science and Information Conference (SAI), IEEE, 2014.

[30] https://www.spinellis.gr/sw/ckjm/

[31] http://www.locmetrics.com

[32] https://activityworkshop.net/software/jarcomp/

[33] P.K Goyal and G. Joshi, *"QMOOD metric sets to assess quality of java program"* in Issues and Challenges in Intelligent Computing Techniques (ICICT), International Conference, IEEE, 2014.

[34] https://logging.apache.org/log4j/

[35] https://github.com

[36] M.A. Hall, *"Correlation-based feature selection for machine learning"*, PhD thesis, The University of Waikato, 1999.

[37] R Development Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/, 2013.

[38] K. Kira and L.A. Rendell, *"The feature selection problem: Traditional methods and a new algorithm"* in AAAI, vol. 2, 1992.

[39] J.P. Lander, *"R for Everyone: Advanced Analytics and Graphics"*, Pearson Education, 2014

[40] H. White, *"Economic prediction using neural networks: The case of IBM daily stock returns",* pp. 451-458, 1988.

[41] T.G. Dietterich, *"An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization"* in Machine Learning, vol. 40, pp. 139-157, 2000.

[42] T.Zhang, *"Adaptive forward-backward greedy algorithm for sparse learning with Linear Models"* in Neural Information Processing Systems Conference, 2008.

[43] C. Hans, *"Elastic net regression modeling with the orthant normal prior"* in Journal of the American Statistical Association, vol. 106.496, pp. 1383-1393, 2011.

[44] L. Brieman, *"Random forests"* in Machine Learning, vol. 45, pp. 5-32, 2001.

LIST OF PUBLICATION

- Accepted paper titled "Evaluation of Feature Selection Techniques for Software Maintenance Prediction" in 2nd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS) to be held in R.V College of Engineering, Bengaluru, Karnataka, India from 21-23 December, 2017.

APPENDIX

Plagiarism Report

SN

ORIGINALITY REPORT

%6	%1	%6	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	KAUR, ARVINDER, and KAMALDEEP KAUR. "STATISTICAL COMPARISON OF MODELLING METHODS FOR SOFTWARE MAINTAINABILITY PREDICTION", International Journal of Software Engineering and Knowledge Engineering, 2013. Publication	%1
2	Inza, I.. "Feature Subset Selection by Bayesian network-based optimization", Artificial Intelligence, 200010 Publication	%1
3	Chandrashekar, Girish, and Ferat Sahin. "In-vivo fault prediction for RF generators using variable elimination and state-of-the-art classifiers", 2012 IEEE International Conference on Systems Man and Cybernetics (SMC), 2012. Publication	%1
4	Al-Jamimi, Hamdi A., and Moataz A. Ahmed. "Machine learning approaches for predicting software maintainability: a fuzzy-based	<%1