A feature selection model for prediction of software defects

Amit Kumar, Yugal Kumar* and Ashima Kukkar

Department of Computer Science and Engineering,
Jaypee University of Information Technology,
Waknaghat, Solan, Himachal Pradesh, India
Email: amitjakhar69@gmail.com
Email: yugalkumar.14@gmail.com
Email: ashi.chd9@gmail.com
*Corresponding author

Abstract: Software is a collection of computer programs written in a programming language. Software contains various modules which make it a complex entity and it can increase the defect probability at the time of development of the modules. In turn, cost and time to develop the software can be increased. Sometimes, these defects can lead to failure of entire software. It will lead to untimely delivery of the software to the customer. This untimely delivery can responsible for withdrawal or cancellation of project in future. Hence, in this research work, some machine learning algorithms are applied to ensure timely delivery and prediction of defects. Further, several feature selection techniques are also adopted to determine relevant features for defect prediction.

Keywords: software; defect; prediction; classifier; feature selection; cognitive weight.

Reference to this paper should be made as follows: Kumar, A., Kumar, Y. and Kukkar, A. (2020) 'A feature selection model for prediction of software defects', *Int. J. Embedded Systems*, Vol. 13, No. 1, pp.28–39.

Biographical notes: Amit Kumar is presently working as an Assistant Professor (Senior Grade) in Department of Computer Science and Engineering at Jaypee University of Information Technology (JUIT), Waknaghat, Himachal Pradesh, India. He has more than six years of teaching and research experience at reputed colleges and universities of India. He completed his PhD in Computer Science and Engineering from Birla institute of Technology, Mesra, Ranchi. His primary area of research includes software engineering and data mining.

Yugal Kumar is currently working as an Assistant Professor (Senior Grade) in Department of Computer Science and Engineering at Jaypee University of Information Technology (JUIT), Waknaghat, Himachal Pradesh, India. He has more than ten years of teaching and research experience at reputed colleges and universities of India. He has completed his PhD in Computer Science and Engineering from Birla Institute of Technology, Mesra, Ranchi. His primary area of research includes meta-heuristic algorithms, data clustering, swarm intelligence, pattern recognition, medical data international journals and conferences of repute. He is serving as editorial review board member of various journals including *Soft Computing*, *Neurocomputing*, *Computer Methods and Programs in Biomedicine*, *PLOSE ONE*, *Journal of Advanced Computational Intelligence and Intelligent Informatics* and *Journal of Information Processing System*.

Ashima Kukkar is a Research Scholar in the Department of Computer Science and Engineering at Jaypee University of Information Technology (JUIT), Waknaghat, Himachal Pradesh, India. His primary area of research includes software engineering and data mining.

This paper is a revised and expanded version of a paper entitled 'A supervised bug report classification with incorporate and textual field knowledge' presented at International Conference on Computational Intelligence and Data Science (ICCIDS), The NorthCap University Gurugram, India, 7–8 April 2018.

1 Introduction

Software defects can occur due to ambiguous and incorrect implementation of code or logic. The defect prediction in software is a challenging task for developers. The defects can occur any stage of development process and must be handled carefully. The IEEE/ANSI standard for software defects errors are:

- an erroneous software due to human mistakes
- the system fails owing to accidental condition of resulting faults to function as required
- a defect is an irregularity in a product
- a failure happens when some functional unit of software could no longer perform its required functionality within its specified time limits.

Software defects are responsible for increasing the development time and maintenance cost. Software defects can lead to degrade the performance of software and also responsible for failure of entire system. Thus, it is important to predict defects prior to delivery rather than at the time of maintenance. So, if any kind of defect occurs, it must be handled immediately. The defect free module increases the reliability, maintainability, usability, and efficiency of software. The development team uses software quality metrics to resolve defects during various phases of software development. The quality of software can be described as intrinsic product quality and customer satisfaction. The software quality can be measured using software metrics and divided into in-process and end-product metrics. The quality software consists of minimal defects, delivered within time frame, meets the requirements and expectations of customers and it should be maintainable. Software quality according to ISO standard is defined as the features and characteristics of a product or service that abides its capability to assure stated and unstated needs. A lot of research work has been done in effort estimation and fault prediction since last 50 years. Hall et al. gave an overview of several defect prediction models based on various studies (Hall et al., 2012). Still, one cannot make an accurate estimation of efforts and defect. These defects can occur in several forms and having dynamic nature. Most of models consider line of code to estimate the efforts (Kearney et al., 1986). McCabe (1976) invented another measure known as cyclomatic complexity (CC). The LOC measure depends on programmer ability, but McCabe disregards the internal structure of the source code. Later, Halstead developed another method to calculate the development time, efforts, difficulty, and errors. The occurrences of operators and operands are used to measure the various parameters of software but do not include the control structure (Halstead, 1977). But, the complexity of source code depends on control structure as well as the internal structure of the software. Hence, both structures should be included for software estimation.

Cognitive informatics (CI) is widely used in artificial intelligence, software engineering and cognitive science to find the solution of many problems (Wang, 2002). The cognitive complexity of software majorly depends on three factors – input, output and architectural flow (Wang, 2004). CI considers both of internal and control structure of the software to measure software prediction. Till date, many classification models have been developed to predict the defected modules automatically. The basic hypothesis with defects prediction is that, the module which is presently developing is faulty if the same module with the related product developed earlier in the similar environment was faulty (Zheng, 2010). The historical data of testing phase are also helpful for risk analysis. The other thing regarding defect prediction is that the testers can focus on the modules, which seems to be defective. This may reduce the development cost and time of the project at the testing phase and lead to a better release. The PROMISE repository published a number of public NASA datasets for software defects (http://PROMISE.ivv.nasa.gov). In this work, CM1, JM1, KC3, MC1, MC2, PC1, PC2, PC3, and PC4 datasets are used for experiments. These datasets contain many metrics related to project, but all are not unique and important. Sometimes, unnecessary features decrease the performance of classifiers. Therefore, feature selection techniques are also important for removing the irrelevant features. These techniques not only find relevant features, but also increase the performance of classifiers. These techniques rank the features of dataset and higher rank features are considered for classification of software defects.

In this work, machine learning technique like: bagging, Bayes net, classification via regression (CVR), J48, JRip, K-star, logistic regression, multilayer perceptron, naive Bayes, and support vector machine are utilised for software fault prediction. Several performance parameters are used to measure the classifiers outcome like: probability of detection (PD), probability of false alarm (PF), accuracy, true negative value (TNR), precision, false negative rate (FNR), F-measure and area under curve (AUC).

Research gap

It is observed that lacks of mechanisms are reported to identify the important features of a program to determine the defects. But, the features have great impact on the performance of classifiers. Some features are more valuable than others and significant features contribute a lot in defect prediction. Further, the cognitive features are not taken into consideration for measuring the defect prediction.

Contribution of work

The main objective of this work is to improve the defect prediction rate.

- to introduce the concept of cognitive features for measuring defect prediction
- to determine the valuable features using different feature selection mechanisms
- to adopt the machine learning techniques for defect prediction.

2 Related work

A lot of work has been done in the field of software defect prediction and determined efforts, development time and errors using cognitive techniques. The previous studies

reported that processed data performs well with historical data (Nagappan et al., 2010; Madeyski and Jureczko, 2010). Shihab et al. suggested that the NASA PROMISE datasets are most popular and used increasingly (Kamei and Shihab, 2016). These datasets have significant impact on the performance of classifiers. So, it is necessary to consider the quality of dataset for prediction models. The datasets contain noise, outliers and missing values; can affect the classifier's performance. Gray et al. found that lack of data cleaning method resulted poor defect prediction rate (Jiang et al., 2009). Another study revealed the importance of data quality (Gray et al., 2012). Therefore, features of the dataset are also taken into consideration to develop a defect prediction model. The feature selection can improve the performance of building model (Shivaji et al., 2009). Arisholm et al. stated that there is a limited impact of classifier techniques in prediction (Arisholm et al., 2010). Kehan et al. focused on the problem of attribute selection in context of software quality estimation (Gao et al., 2011). The proposed work is divided into two phases. In first phase, feature ranking techniques were applied on the dataset and select the best $\log_2 \lceil n \rceil$ attributes from the datasets. In second phase, the CR-based subset selection techniques were applied, i.e., exhaustive search, heuristic search, and automatic hybrid search. This study showed that the feature ranking technique provide similar performance. But, hybrid search algorithm performs better than CR-based techniques. Hall et al. examined 208 studies and found that defect prediction performance varies between studies (Hall et al., 2012). Jakhar and Rajnish (2016) showed the affect of cognitive estimation on development time, efforts, and errors for software defect prediction. The process can be divided into two phases. In first phase, a new cognitive approach is developed based on goodness and correlation is calculated between proposed cognitive approach and Halstead measure. In second phase, the machine learning techniques are implemented on both of original and derived dataset. The various performance parameters are applied to evaluate performance of machine learning algorithms and it is found that cognitive dataset gives better results than original dataset. Catal and Diri (2009) adopted machine learning algorithm for fault prediction. The different metrics are used in this work. This work mainly focuses on reliability and effectiveness of algorithms for fault prediction. The machine learning techniques were implemented on nine public dataset from NASA's PROMISE repository ((http://PROMISE.ivv.nasa.gov). Aggarwal and Tomar (2014) proposed a new feature selection technique-based linear twin support vector machine model to predict the software defects. A new F-score-based feature selection technique is proposed to select significant feature on four dataset from NASA's PROMISE repository (Kamei and Shihab, 2016). Park et al. (2013) designed polynomial function-based neural network predictors for detection of software defects. In this work, several rules are developed and are expressed in the form of 'if-then'. The polynomial function divides into two categories, one is linear function and another is quadratic function-based neural networks. The ROC parameter is used to analyse the performance of the model. Jakhar and Rajnish (2016) suggested a cognitive approach for measuring complexity, development time and understandability of a program (Jakhar and Rajnish, 2014). In this work, a new concept of weighted method complexity (NWMC), understandability (UA) and development time was proposed. Menzies and Greenwald (2007) developed a prediction technique based on Dempster-Shafer evidence theory and information fusion to determine software defect. Several other researches also focuses on the prediction in various field with relevant features among the features set and shown that the results get improved (Arisholm et al., 2010; Jakhar and Rajnish, 2018; Htike, 2018; Priya and Sherly, 2018; Harikumar et al., 2017).

3 Classification, feature ranking and cognitive techniques

Machine learning is a process through which a system improves its performance from past experience. Machine learning is of three types: supervised, unsupervised, ensembles. In this paper, bagging, Bayes net, CVR, J48, JRip, KStar, logistic regression, multilayer perceptron, naive Bayes, sequential minimal optimisation and support vector machine along with the feature ranking techniques: chi squared, gain ratio, information gain, and symmetrical uncertainty attribute evaluation are used for defect prediction (Cameron and Tivedi, 1998; Witen and Frank, 2005; Pearson, 1901).

3.1 Classification techniques

In classification, machine learns from the training data or labelled data. In case of software defect prediction, attributes are labelled as defective or not defective and the classifier predicts on the testing data that which one of the instance is defected.

3.1.1 Bagging

Bagging stands for bootstrap aggregation (Breiman, 1996). It is an ensemble learning technique. It combines more than one model to make prediction or decision or do classification. Suppose a dataset S is provided to the classifier with t tuples and it has to do i iterations where, i = 1, 2, … m, which is correspond to the m models that it has to create. A set, S_i, will be created of t tuples sampled with replacement from the original set of tuples in S.

3.1.2 Logistic regression (LR)

It uses the log-likelihood method which can be expressed in equation (1) (de Mantaras and Armengol, 1998):

$$\sum_{i-1}^{n}(1-x^{(i)})\log\left(1-\Pr\left[1|\ a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}\right]\right) + x^{(i)}\log\left(\Pr\left[1|\ a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}\right]\right) \quad (1)$$

where $x(i)$ can either be 0 or 1. Where other parameters are measured with the help of equation (2):

$$\Pr\left[1\,|\,a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(1)} - k\right]\Big) = \frac{1}{1+\exp\left(1-w_0-w_1a_1-\cdots-w_ka_k\right)} \quad (2)$$

It is the transformed variable which is approximated using linear function. The weight W_i must be chosen in such a way that it maximises the log-likelihood function. It generalises the data in the short dimension especially in between 0 and 1, such that each instance can be correctly classified and it also has a linear decision boundary. That is why it is also known as logit or logistic model.

3.1.3 Classification via regression

The CVR technique consists of a sub-process. The sub-process has a regression learner which generates the regression model. Based on the regression model a classification model was built or it will be used to train the classier model (Frank et al., 1998).

3.1.4 JRip

It is proposed by William W. Cohen and implements a propositional rule learner, repeated incremental pruning to produce error reduction (RIPPER) (Cohen, 1995). It has two stages

1 Building stage: it iteratively grows and prunes the rules until the error rate is 50%. It has two phases
   - Growing phase: each rule is generated greedily by adding conditions until the rule become 100% correct.
   - Prune phase: incrementally prune each rule and allows the pruning of all final sequences p of the condition. The pruning metric is defined as equation (3):

$$\frac{p}{p+n} \quad (3)$$

2 Optimisation stage: after generating initial rule set, say R, generate and prune two variants of each rule R from randomised data using procedure 1(a) and 1(b). One of the variants is generated using greedily adding condition to the rule set and one is generated from the rule set. The description length is computed for both of the variants and the original rule set. The variant with the smallest possible description length is used as the representative of the R in the rule set.

3.1.5 J48

J48's are generated by recursive partitioning. Recursive partitioning means repeatedly splitting on the values of attributes. In every recursion the algorithm follows the following steps (Korting, 2006):

- It uses gain ratio or information gain or can be some other attribute ranking technique to determine at what attribute the tree has to be split.
- Instances in the dataset are arranged in subsets, one for each value of the attribute selected to split the tree in case of a nominal attribute. In case of numerical attributes, subsets are formed for disjoint ranges of attribute values.
- A tree is returned with one edge or branch for each subset. Each branch has a descendant sub tree or a label value produced by applying the same algorithm recursively.

In general, the recursion stops when subset becomes pure or when most of the instances have same label value. This is a generalisation of the first approach; with some error threshold. However there are other halting conditions.

3.1.6 Multilayer perceptron

MLPs (Haykin, 1998) it is a mimic of artificial nervous system. It is a supervised learning technique in which to train the network back propagation is used. There are multiple nodes those are connected together and the input which is provided to them is the threshold weighted sum of the previous layer. The threshold weighted sum which is mapped is known as activation function. Mainly there are two kinds of activation function h are used and both of them are sigmoid, and can be described in equation (4):

$$y(v_i) = \tanh(v_i) \quad \text{and} \quad y(v_i) = \left(1+e^{-v_i}\right)^{-1} \quad (4)$$

Here, the first function is hyperbolic tangent function whose range is from –1 to 1 and the second one is logistic function whose range is from 0 to 1. In this work, two hidden layer is used to implement multi-layer perceptron.

3.1.7 Lib support vector machines

LibSVM (Chan and Lin, 2001) it is an open source machine learning libraries, developed by National Taiwan University and written in C++. It uses the SMO algorithm for kernelised support vector machine, supporting both classification and regression.

3.1.8 Sequential minimal optimisation

SMO (Platt, 1998) is an algorithm which is used for training the support vector machine when quadratic programming problem arises. It is implemented by the LIBSVM tools (Chan and Lin, 2001; Zanni, 2006). It is an iterative

algorithm which is used to solve the optimisation problem. It breaks the problem into two sub problems, which are then solved analytically.

3.1.9 Naive Bayes

In machine learning naive Bayes (John and Langley, 1995) is a family of simple probabilistic estimator which is based on Bayes theorem with an assumption that the features are independent. It is a condition probabilistic model for a problem to be classified represented by a vector $x = (x_1, \ldots, x_n)$ which represent n features given in the problem, it assign these instances to the class for each of k possible outcomes or classes C_k (Murty and Devi, 2011).

$$\left(C_k \mid x_1, \ldots, x_n\right) \tag{5}$$

This approach is not feasible when a dataset has a large number of instances so to make it more tractable we use Bayes theorem as defined in equation (6):

$$p\left(C_k|x\right) = p\left(C_k\right) p\left(x|\mathrm{C}_k\right) \big/ p(x) \tag{6}$$

where $p(C_k|x)$ is posterior, $p(C_k)$ is prior, $p(x|C_k)$ is likelihood, and $p(x)$ is evidence.

3.1.10 K-star

K* is an instance-based classifier, i.e., the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function. It uses an entropy-based distance function (Cleary and Trigg, 1995).

3.2 Cognitive techniques

It is one of the ways by which we can estimate the efforts put by a human to develop or understand the software.

The cognitive techniques that are used in this paper:

1 *NWMC:* it is known as new weighted method complexity (Jakhar and Rajnish, 2014). It can be used to measure the difficulty level and increase the quality by using several important parameters of the source code. The formula is given in equations (7) and (8).

$$NWMC = N_p, W_c \tag{7}$$

where

$$N_P = \left(N_i + N_o + N_{lp} + N_{fp}\right) \tag{8}$$

where

N_i individual number of input

N_o individual number of output

N_{lp} number of local parameters

N_{fp} number of formal parameters.

2 *Understandability:* it is denoted as UA (Jakhar and Rajnish, 2014). It is a quality factor which can be used to understand the code before testing and maintenance. It is also used to identify the relationship between NWMC and difficulty which is known as level of understandability, and it is described in equation (9).

$$UA = (NWMC)^a \times b \tag{9}$$

where a and b are constants, $a = 0.48$, and $b = 0.62$.

3 *Development time:* it is the time taken by the software developer to develop the product, and the formula is given in equation (10) (Jakhar and Rajnish, 2014).

$$Dev\_Time = a + (b \times NWMC) \tag{10}$$

The values of a and b are calculated by the help of regression and values for development time are as: $a = 0.9793$, and $b = 0.0964$.

4 *Calculated efforts:* it is the estimation of required efforts to develop the software (Jakhar and Rajnish, 2016). The equation which calculates the efforts is provided in equations (11) and (12).

$$\begin{aligned} Cal\_Effort = \big((&No.\_of\_operands \\ &\times No.\_of\_operators) \\ &+ \left(Branch\_count \times W_c\right) + LOC\_Exe\big) \end{aligned} \tag{11}$$

If the branch count < 2 then W_c is 2 otherwise, W_c is chosen as 3.

$$W_c = \begin{Bmatrix} 2, & \text{if } (Branch\_count < 2) \\ 3, & \text{otherwise} \end{Bmatrix} \tag{12}$$

5 *Development time:* with the help of calculated effort, the time required to develop a software is calculated as given in equation (13) (Jakhar and Rajnish, 2016):

$$Cal\_Dev\_Time = (Cal\_Effort)/10 \tag{13}$$

6 *Errors:* equation (21) is used to calculate the number of errors occurs in the software (Jakhar and Rajnish, 2016).

$$Number\_of\_Errors = (Cal\_Effort)^{0.2} - 1.5 \tag{14}$$

4 Performance assessment parameters

To validate the work of this paper, several performance parameters are used. The confusion matrix is used to calculate various parameters like: sensitivity, specificity, precision, false alarm, false negative rate, accuracy and receiver operating characteristics (ROC) curve, given in Table 1.

1 True positive (TP): the number of instances which are positive and classified by the classifier as positive.

2 False positive (FP): the number of modules which are not defective but predicted as defective.

3 True negative (TN): those instances which are negative and correctly classified by the classifier as negative.

4 False negative (FN): those modules which have defects but predicted by the classifier as not defective.

Table 1 Confusion matrix

Predicted class	*Defected modules in logs*	
	Defective	*Not defective*
Defective	True positive (TP)	False positive (FP)
Not defective	False negative (FN)	True negative (TN)

5 Experiment description

This section describes about the definition of the experiment, datasets, what metrics we have selected, design of the experiment, tools, and validation evaluation. The system which is used in this study must have following features:

1 the development of the system must be done by a group not by an individual

2 it must developed by professionals not by students

3 it should not be developed in an artificial environment but in industrial environment

4 it should be large enough to be comparable to industrial projects (Khoshgoftaar et al., 2006).

The construct validity refers to the degree to which the dependent and independent variables in the study measure what they claim to measure (Pai and Dugan, 2007).

5.1 Experiment definition

The objective of this work is to identify the best software fault prediction model with different group of metrics. The experiment is performed on NASA's PROMISE repository ((http://PROMISE.ivv.nasa.gov). The effectiveness and reliability of the proposed model is identified by using various performance parameters.

5.2 Datasets characteristics

The nine public datasets, namely CM1, JM1, MC1, MC2, PC1, PC2, PC3, PC4, and KC3 are used for performance evaluation of machine learning techniques. These datasets are downloaded from NASA's PROMISE repository ((http://PROMISE.ivv.nasa.gov). The description of each dataset is given in Table 2.

Table 2 Datasets description

Name of dataset	*Number of modules*	*Defects (%)*
CM1	505	9.5%
JM1	10,885	19.3%
MC1	9,466	0.7%
MC2	161	32.29%
PC1	1,109	6.94%
PC2	5,589	0.4%
PC3	1,563	10.2%
PC4	1,548	12.2%
KC3	458	9.3%

5.3 Design of the experiments

In this work, the attribute named as defective which is a dependent variable used to predict the percentage of the defect and those which are not defective based on the prediction model. There are four experiments performed in this work. The metrics used in these experiments are described as follows:

Experiment 1 All metrics are considered from the original data and then we apply the classification techniques.

Experiment 2 Cognitive techniques are applied to the datasets and only cognitive and those attributes that derived from these cognitive techniques.

Experiment 3 Only result of first three cognitive techniques described in Section 3.3 are embedded in the original datasets to perform the classification.

Experiment 4 The original datasets are considered, then perform the feature selection, we start by selecting four attributes up to 23 attributes, to predict the effect on the software defect prediction using classification approach and try to validate whether the statement claimed is applicable to every dataset or depends on the size of datasets only (Gao et al., 2011).

5.4 Tools

In this work, NASA's PROMISE repository which is in attribute-relation file format (ARFF) format, which can be easily open in Waikato environment for knowledge analysis (WEKA) is used to perform experiments (http://PROMISE.ivv.nasa.gov).

6 Results evaluation

This section describes the outcome of the experiment that was described in previous section. All the experiments in this section were implemented using eleven aforementioned classification and feature ranking algorithms. The performance of the classifier is compared by using various performance parameters but the preference is given more to the area under receiver operating characteristics curve (ROC). The overall results of all the experiments are given in Tables 3–12 and shown in the Figures 1–4.

6.1 Experiment 1

In Experiment 1, all features of the original dataset are considered and aforementioned machine learning techniques are applied for software defect prediction. The performance parameters are computed to evaluate the performance of each classifier with each dataset. The results of aforementioned machine learning techniques with all datasets using accuracy, precision and ROC are reported in Tables 3–5.

Table 3 Accuracy rate of Experiment 1

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.82	0.71	0.82	0.80	0.82	0.78	0.82	0.82	0.81	0.81	0.82
JM1	0.90	0.70	0.91	0.88	0.91	0.86	0.91	0.88	0.88	0.86	0.91
KC3	0.99	0.87	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.94	0.99
MC1	0.71	0.58	0.69	0.71	0.71	0.65	0.67	0.65	0.60	0.69	0.75
MC2	0.92	0.64	0.92	0.86	0.92	0.86	0.91	0.91	0.86	0.88	0.92
PC1	0.99	0.90	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.99
PC2	0.88	0.73	0.87	0.85	0.88	0.87	0.91	0.89	0.85	0.52	0.90
PC3	0.89	0.74	0.89	0.86	0.86	0.82	0.87	0.91	0.89	0.85	0.88
PC4	0.99	0.90	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.99

Table 4 Precision of Experiment 1

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.88	0.92	0.88	0.87	0.87	0.89	0.87	0.87	0.87	0.92	0.87
JM1	0.55	0.32	0.57	0.43	0.55	0.38	0.74	0.56	0.47	0.45	1.00
KC3	0.91	0.97	0.92	0.94	0.92	0.91	0.91	0.92	0.92	0.93	0.91
MC1	0.99	1.00	0.99	0.99	0.99	1.00	0.99	0.99	0.99	1.00	0.99
MC2	0.74	0.80	0.69	0.80	0.78	0.71	0.67	0.70	0.72	0.72	0.73
PC1	0.93	0.94	0.92	0.92	0.92	0.93	0.92	0.93	0.92	0.94	0.92
PC2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PC3	0.91	0.96	0.91	0.91	0.90	0.92	0.91	0.91	0.91	0.98	0.90
PC4	0.92	0.94	0.90	0.91	0.90	0.89	0.88	0.92	0.93	0.89	0.88

Table 5 ROC of Experiment 1

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.72	0.68	0.68	0.50	0.50	0.65	0.50	0.51	0.54	0.73	0.50
JM1	0.72	0.70	0.70	0.67	0.54	0.59	0.52	0.69	0.70	0.67	0.50
KC3	0.83	0.77	0.70	0.69	0.57	0.52	0.50	0.70	0.73	0.82	0.50
MC1	0.93	0.91	0.88	0.74	0.65	0.76	0.62	0.89	0.93	0.91	0.50
MC2	0.57	0.66	0.64	0.70	0.67	0.57	0.50	0.44	0.60	0.69	0.63
PC1	0.74	0.67	0.78	0.45	0.50	0.62	0.50	0.80	0.84	0.69	0.50
PC2	0.79	0.88	0.78	0.50	0.52	0.64	0.50	0.79	0.89	0.84	0.50
PC3	0.79	0.79	0.83	0.67	0.53	0.72	0.55	0.82	0.69	0.80	0.50
PC4	0.91	0.79	0.91	0.76	0.63	0.75	0.52	0.90	0.88	0.80	0.53

6.2 Experiment 2

In Experiment 2, the development effort, time and number of errors are calculated with the help of proposed cognitive technique. The outcome of the equations (11) to (14), i.e., development effort, time and number of errors are correlated with the outcome of Halstead measure. The graph of correlation result is shown in Figure 1. The correlation result indicates a good relation exists between the proposed and Halstead measure. In this experiment, two types of attributes are considered to evaluate the performance of proposed cognitive techniques. These attributes are cognitive attributes and cognitive techniques derived attributes. The simulation results using all datasets are demonstrated in Tables 6–8.

Table 6 Accuracy rate of Experiment 2

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.87	0.87	0.87	0.87	0.87	0.85	0.87	0.87	0.85	0.83	0.87
JM1	0.81	0.72	0.81	0.81	0.81	0.80	0.82	0.81	0.81	0.81	0.82
KC3	0.91	0.71	0.91	0.91	0.87	0.87	0.91	0.91	0.91	0.88	0.91
MC1	0.99	0.93	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.95	0.99
MC2	0.67	0.40	0.65	0.73	0.69	0.56	0.67	0.63	0.65	0.67	0.67
PC1	0.92	0.67	0.92	0.92	0.92	0.91	0.91	0.93	0.92	0.87	0.92
PC2	1.00	0.88	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.97	1.00
PC3	0.89	0.72	0.90	0.90	0.90	0.86	0.90	0.89	0.90	0.17	0.90
PC4	0.87	0.63	0.87	0.87	0.87	0.84	0.87	0.87	0.88	0.85	0.87

Table 7 Precision of Experiment 2

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.87	0.87	0.88	0.87	0.87	0.87	0.87	0.87	0.86	0.90	0.87
JM1	0.83	0.88	0.83	0.83	0.83	0.83	0.82	0.82	0.83	0.84	0.82
KC3	0.91	0.93	0.92	0.91	0.91	0.90	0.91	0.91	0.91	0.93	0.91
MC1	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.99
MC2	0.70	0.80	0.67	0.72	0.72	0.64	0.67	0.67	0.67	0.69	0.67
PC1	0.93	0.99	0.92	0.92	0.92	0.94	0.92	0.92	0.92	0.95	0.92
PC2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PC3	0.90	0.94	0.90	0.90	0.90	0.92	0.90	0.90	0.90	0.92	0.90
PC4	0.87	0.91	0.87	0.87	0.87	0.90	0.87	0.88	0.88	0.88	0.87

Table 8 ROC of Experiment 2

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.71	0.50	0.78	0.50	0.50	0.42	0.50	0.73	0.66	0.70	0.50
JM1	0.71	0.71	0.72	0.66	0.54	0.62	0.51	0.71	0.72	0.66	0.50
KC3	0.77	0.68	0.74	0.50	0.55	0.61	0.50	0.77	0.74	0.74	0.50
MC1	0.94	0.90	0.91	0.69	0.65	0.93	0.65	0.86	0.87	0.86	0.50
MC2	0.50	0.53	0.60	0.58	0.59	0.42	0.50	0.56	0.50	0.64	0.50
PC1	0.77	0.79	0.80	0.50	0.50	0.74	0.52	0.75	0.77	0.80	0.50
PC2	0.72	0.72	0.77	0.50	0.50	0.63	0.50	0.52	0.78	0.71	0.50
PC3	0.75	0.74	0.72	0.50	0.50	0.56	0.51	0.74	0.74	0.69	0.50
PC4	0.71	0.64	0.68	0.50	0.50	0.67	0.50	0.70	0.68	0.66	0.51

6.3 Experiment 3

In Experiment 3, a proposed weighted method complexity (WMC) is used along with the proposed technique of Section 6.8.1 of Experiment 2 to design the relevant features dataset. In this experiment, only selected features are considered for performance evaluation of above-said machine learning techniques. The experimental results using all datasets are mentioned in Tables 9–11.

6.4 Experiment 4

In Experiment 4, the original dataset are taken into account and rank all features of the dataset using aforementioned feature ranking algorithms. A new dataset is created using higher rank features. The new dataset consists of four features out of twenty four features. The machine learning techniques are applied on newly created dataset for software defect prediction. The simulation results of these techniques using all datasets are reported in Table 12. The graphical comparison of correlation, accuracy, precision and ROC parameters are depicted in Figures 1–4. Figure 1 shows the comparison of Halstead and proposed cognitive technique using development time, effort and number of errors. The comparison of accuracy parameter of all four experiments is given in Figure 2. The comparison of precision and ROC parameters are mentioned in Figures 3–4.

Table 9 Accuracy rate of Experiment 3

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.87	0.87	0.87	0.87	0.87	0.84	0.87	0.86	0.87	0.84	0.87
JM1	0.81	0.71	0.81	0.81	0.81	0.81	0.82	0.81	0.81	0.81	0.82
KC3	0.91	0.71	0.91	0.91	0.87	0.89	0.91	0.91	0.90	0.90	0.91
MC1	0.99	0.88	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.99
MC2	0.69	0.40	0.65	0.73	0.69	0.52	0.67	0.65	0.67	0.67	0.67
PC1	0.92	0.68	0.92	0.92	0.92	0.92	0.91	0.93	0.92	0.88	0.92
PC2	1.00	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00
PC3	0.89	0.70	0.90	0.90	0.90	0.86	0.90	0.90	0.90	0.13	0.90
PC4	0.87	0.68	0.87	0.87	0.87	0.85	0.87	0.87	0.87	0.85	0.87

Table 10 Precision of Experiment 3

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.87	0.87	0.88	0.87	0.87	0.87	0.87	0.87	0.88	0.91	0.87
JM1	0.83	0.88	0.83	0.83	0.83	0.83	0.82	0.82	0.83	0.83	0.82
KC3	0.91	0.93	0.92	0.91	0.91	0.90	0.91	0.91	0.91	0.93	0.91
MC1	1.00	1.00	0.99	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99
MC2	0.71	0.80	0.67	0.72	0.71	0.62	0.67	0.67	0.68	0.69	0.67
PC1	0.93	0.99	0.92	0.92	0.92	0.94	0.92	0.92	0.92	0.95	0.92
PC2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PC3	0.90	0.95	0.90	0.90	0.90	0.91	0.90	0.90	0.90	0.82	0.90
PC4	0.88	0.91	0.87	0.87	0.87	0.89	0.87	0.87	0.88	0.87	0.87

Table 11 ROC of Experiment 3

	BAGGING	*BN*	*CVR*	*J48*	*JRIP*	*K-star*	*LS*	*LR*	*MLP*	*NB*	*SMO*
CM1	0.71	0.50	0.78	0.50	0.50	0.42	0.50	0.71	0.75	0.70	0.50
JM1	0.73	0.71	0.72	0.66	0.55	0.65	0.51	0.71	0.72	0.66	0.50
KC3	0.77	0.67	0.74	0.50	0.55	0.60	0.50	0.76	0.75	0.76	0.50
MC1	0.94	0.90	0.91	0.69	0.65	0.93	0.65	0.87	0.86	0.86	0.50
MC2	0.52	0.53	0.58	0.58	0.58	0.33	0.50	0.56	0.60	0.65	0.50
PC1	0.77	0.78	0.80	0.50	0.50	0.71	0.52	0.77	0.77	0.79	0.50
PC2	0.72	0.72	0.77	0.50	0.50	0.63	0.50	0.63	0.77	0.76	0.50
PC3	0.75	0.74	0.71	0.50	0.50	0.64	0.51	0.75	0.75	0.69	0.50
PC4	0.69	0.63	0.66	0.50	0.50	0.64	0.50	0.68	0.65	0.67	0.51

Table 12 Best results of all performance parameters of Experiment 4

Datasets	*Accuracy*	*TPR*	*TNR*	*FPR*	*FNR*	*Precision*	*ROC*
CM1	0.88	1.00	0.62	0.38	0.00	0.92	0.81
JM1	0.82	0.51	1.00	0.00	0.49	0.82	0.74
KC3	0.92	1.00	0.77	0.23	0.00	0.97	0.85
MC1	1.00	1.00	0.75	0.25	0.00	1.00	0.97
MC2	0.75	1.00	0.75	0.25	0.00	0.80	0.72
PC1	0.92	1.00	0.53	0.47	0.00	0.94	0.84
PC2	1.00	1.00	0.43	0.57	0.00	1.00	0.77
PC3	0.91	1.00	0.88	0.13	0.00	0.98	0.85
PC4	0.90	1.00	0.83	0.17	0.00	0.97	0.92

Source: Jakhar and Rajnish (2016)

Figure 1 Correlation results of Halstead and proposed cognitive technique (see online version for colours)

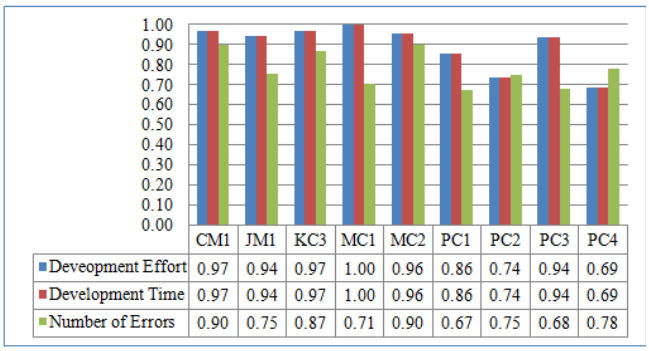


Figure 2 Accuracy rate of all experiments (see online version for colours)

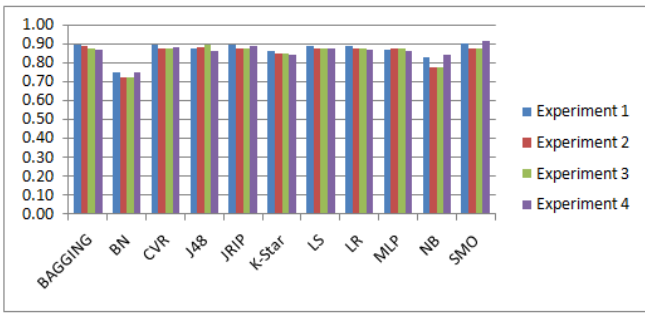


Figure 3 Precision rate of all experiments (see online version for colours)

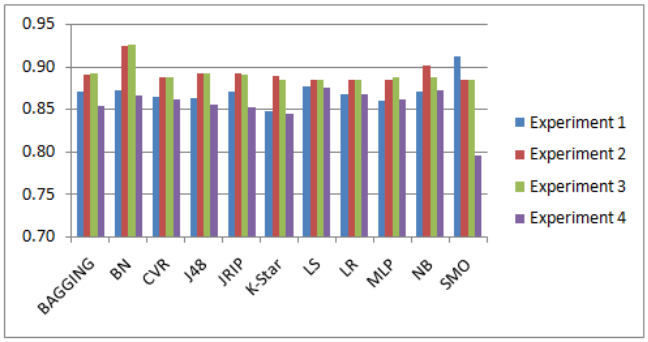
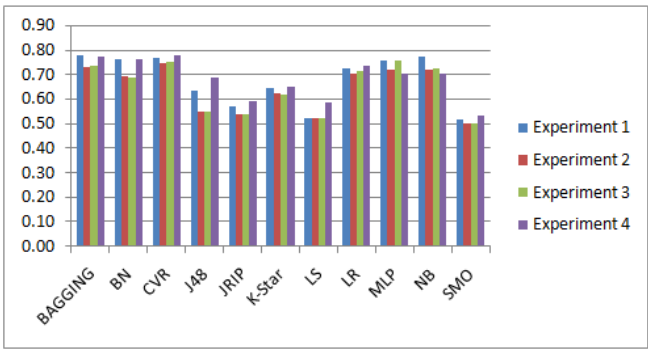


Figure 4 ROC rate of all experiments (see online version for colours)



7 Conclusions

In this work, eleven classification algorithms are applied to predict the software defects. To evaluate the performance of these classification algorithms, four experiments are considered. In addition to it, several feature selection techniques are adopted to determine relevant features for software defects prediction. Further, the ten cross fold technique is used to measure the performance of all machine learning algorithms. The accuracy, precision and ROC parameters are used to compare the performance of all machine learning techniques using four experiments. In this study, a new dataset is also developed on the basis of most relevant features. These features are determined using feature selection techniques. It is observed that feature selection techniques improve the classification results of machine learning algorithm as well as defect prediction rate. The results of the all experiments demonstrate that the proposed models have enhanced the capability of the machine learning algorithms in several aspects.

References

Aggarwal, S. and Tomar, D. (2014) 'A feature selection based model for software defect prediction', *International Journal of Advance Science and Technology*, Vol. 65, No. 4, pp.39–58.

Arisholm, E., Briand, L.C. and Johannessen, E.B. (2010) 'A systematic and comprehensive investigation of methods to build and evaluate fault prediction models', *Journal of Systems and Software*, Vol. 83, No. 1, pp.2–17.

Breiman, L. (1996) 'Bagging predictors', *Machine Learning*, Vol. 24, No. 2, pp.123–140.

Cameron, A.C. and Tivedi, P.K. (1998) *Regression Analysis of Count Data*, Cambridge University Press, Cam-bridge.

Catal, C. and Diri, B. (2009) 'Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem', *Information Sciences*, Vol. 179, No. 8, pp.1040–1058.

Chan, C-C. and Lin, C.J. (2001) 'LIBSVM – a library for support vector machine', *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, No. 3 [online] http://www. csie. ntu. edu. tw/~ cjlin/libsvm.

Cleary, J.G. and Trigg, L.E. (1995) 'K*: an instance-based learner using an entropic distance measure', *12th International Conference on Machine Learning*, pp.108–114.

Cohen, W.W. (1995) 'Fast effective rule induction', *12th International Conference on Machine Learning*, pp.115–123.

de Mantaras, R.L. and Armengol, E. (1998) 'Machine learning from example: inductive and lazy methods', *Data Knowl. Engg.*, Vol. 25, Nos. 1–2, pp.99–123.

Frank, E., Witten, I.H., Wang, Y., Inglis, S. and Holmes, G. (1998) 'Using model trees for classification machine learning', *Machine Learning*, Vol. 32, No. 1, pp.63–76.

Gao, K., Khoshgoftaar, T.M., Wang, H. and Seliya, N. (2011) *Choosing Software Metrics for Defect Prediction: An Investigation on Feature Selection Techniques*, Vol. 41, No. 5, pp.4579–25606, Wiley Online Library.

Gray, D., Bowes, D., Davey, N., Sun, Y. and Christianson, B. (2012) 'Reflections on the NASA MDP datasets', *IET Software*, Vol. 6, No. 6, pp.549–558.

Hall, T., Beecham, S., Bowes, D., Gray, D. and Counsell, S. (2012) 'A systematic literature review on fault prediction performance in software engineering', *IEEE Transactions on Software Engineering*, Vol. 38, No. 6, pp.1276–1304.

Halstead, M.H. (1977) *Elements of Software Science*, Elsevier, New York.

Harikumar, S., Dilipkumar, D.U. and Kaimal, M.R. (2017) 'Efficient attribute selection strategies for association rule mining in high dimensional data', *International Journal of Computational Science and Engineering*, Vol. 15, Nos. 3–4, pp.201–213.

Haykin, S. (1998) *Neural Network: A Comprehensive Foundation*, 2nd ed., Englewood Cli s, NJ, Prentice-Hall.

Htike, K.K. (2018) 'Predicting rainfall using neural nets', *International Journal of Computational Science and Engineering*, Vol. 17, No. 4, pp.353–364.

Jakhar, A.K. and Rajnish, K. (2014) 'Measuring complexity, development time and un-derstandability of a program: a cognitive approach', *I. J. Information Technology and Computer Science*, Vol. 12, No. 7, pp.53–60.

Jakhar, A.K. and Rajnish, K. (2016) 'Cognitive estimation of development e orts, time, errors, and the defects of software', *Walailak J. Sci. and Tech*., Vol. 13, No. 6, pp.465–478.

Jakhar, A.K. and Rajnish, K. (2018) 'Software fault prediction with data mining techniques by using feature selection based models', *International Journal on Electrical Engineering and Informatics*, Vol. 10, No. 3, pp.447–465.

Jiang, Y., Lin, J., Cukic, B. and Menzies, T. (2009) 'Variance analysis in software fault prediction models', *SSRE 2009, 20th International Symposium on Software Reliability Engineering*, IEEE Computer Society, Mysuru, Karnataka, India, 16–19 November, pp.99–108.

John, G.H. and Langley, P. (1995) 'Estimating continuous distributions in Bayesian classifiers', *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp.338–345.

Kamei, Y. and Shihab, E. (2016) 'Defect prediction: Accomplishments and future challenges', *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 5, pp.33–45. doi:10.1109/ SANER.2016.56.

Kearney, J.P., Sedlmeyer, R.L., Thompson, W.B., Gary, M.A. and Adler, M.A. (1986) 'Software complexity measurement', *Comm. ACM*, Vol. 29, No. 11, pp.1044–1050.

Khoshgoftaar, T.M., Seliya, N. and Sundaresh, N. (2006) 'An empirical study of predicting software faults with case-based reasoning', *Software Quality Journal*, Vol. 14, No. 2, pp.85–111.

Korting, T.S. (2006) *C4.5 Algorithm and Multivariate Decision Trees*, Image Processing Division, National Institute for Space Research, INPE Sao Jose dos Campos, SP, Brazil.

Madeyski, L. and Jureczko, M. (2015) 'Which process metrics can significantly improve defect prediction models? An empirical study', *Software Quality Journal*, Vol. 23, No. 3, pp.393–422, doi:10.1007/s11219-014-9241-7.

McCabe, T.J. (1976) 'A complexity measure', *IEEE Trans. Software Engg.*, Vol. 2, No. 4, pp.308–320.

Menzies, T. and Greenwald, J. (2007) 'Data mining static code attributes to learn defect predictors', *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp.2–13.

Murty, M.N. and Devi, V.S. (2011) *Pattern Recognition: An Algorithmic Approach*, Springer Science & Business Media, ISBN 0857294946.

Nagappan, N., Zeller, A., Zimmermann, T., Herzig, K. and Murphy, B. (2010) 'Change bursts as defect predictors', *IEEE 21st International Symposium on Software Reliability Engineering*, pp.309–318.

NASA IV&V Facility (2006) *Metric and Data Program* [online] http://PROMISE.ivv.nasa.gov (accessed December).

Pai, G.J. and Dugan, J.B. (2007) 'Empirical analysis of software fault content and fault proneness using Bayesian methods', *IEEE Transactions on Software Engineering*, Vol. 33, No. 10, pp.675–686.

Park, B-J., Oh, S-K. and Pedrycz, W. (2013) 'The design of polynomial function-based neural network predictor for detection of software defects', *Information Sciences*, Vol. 229, pp.40–57.

Pearson, K. (1901) 'On lines and planes of closest fit to systems of points in space', *Philosophical Magazine*, Vol. 2, No. 11, pp.559–572.

Platt, J. (1998) *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, Microsoft Research, Technical Report MSR-TR-98-14.

Priya, R. and Sherly, E. (2018) 'Fault detection and behavioural prediction of a constrained complex system using cellular automata', *International Journal of Computational Science and Engineering*, Vol. 17, No. 4, pp.411–421.

Shivaji, S., Whitehead, E.J., Akella, R. and Sunghun, K. (2009) 'Reducing features to improve bug prediction', *24th IEEE/ACM International Conference on Automated Software Engineering, ASE '09*, pp.600–604.

Wang, Y. (2002) 'On the cognitive informatics', *Proceedings of the 1st IEEE International Conference on Cognitive Informatics*, Calgary, Alta, Canada, pp.34–42.

Wang, Y. (2004) 'On the cognitive informatics', *Proceedings of the 3rd IEEE International Conference on Cognitive Informatics*, Victoria, BC, Canada, pp.34–42.

Witen, I.H. and Frank, E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed., Morgan Kaufmann, Los Altos, CA.

Zanni, L. (2006) 'Parallel software for training large scale support vector machines on multiprocessor systems', *Journal of Machine Learning Research*, July, Vol. 7, pp.1467–1492.

Zheng, J. (2010) 'Cost-sensitive boosting neural networks for software defect prediction', *Expert Systems with Applications*, Vol. 37, No. 6, pp.4537–4543.