Software Fault Prediction with Data Mining Techniques by Using Feature Selection Based Models

Amit Kumar Jakhar and Kumar Rajnish

Department of Computer Science and Engineering, Birla Institute of Technology,
Mesra, Ranchi, Jharkhand, India
amitkumar.cs08@pec.edu.in, krajnish@bitmesra.ac.in

Abstract: Software engineering activities comprise of several activities to ensure that the quality product will be achieved at the end. Some of these activities are software testing, inspection, formal verification and software defect prediction. Many researchers have been developed several models for defect prediction. These models are based on machine learning techniques and statistical analysis techniques. The main objective of these models are to identify the defects before the delivery of the software to the end user. This prediction helps project managers to effectively utilize the resources for better quality assurance. Sometimes, a single defect can cause the entire system failure and most of the time they drop the quality of the software system drastically. Early identification of defects can also help to make a better process plan which can handle the defects effectively and increase the customer satisfaction level. But the accurate prediction of defects in software is not an easy task because this is an indirect measure. Therefore, it is important to find suitable and significant measures which are most relevant for finding the defects in the software system. This paper presents a feature selection based model to predict the defects in a given software module. The most relevant features are extracted from all features with the help of seven feature selection techniques and eight classifiers are used to classify the modules. Six NASA software engineering defects prediction data sets are used in this work. Several performance parameters are also calculated for measuring the performance and validation of this work and the results of the experiments revealed that the proposed model has more capability to predict the software defects.

Keywords: software fault prediction, classification techniques, feature selection, f-measure, area under curve

1. Introduction

Defects mainly occurs due to poor coding practices, which can further lead to the poor quality of the software system and high maintenance cost are imposed when the product is delivered to the customer and it also increases customer dissatisfaction [1, 14]. Till now many classification models have been developed and they can automatically predict the defected modules of the software before the testing commences. Basically, these models use different software metrics of earlier product releases and collect the data of testing phases. The basic hypothesis regarding 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 [3]. The collected data of the testing phase of the previous completed projects are also helpful for risk analysis. Presence of the defects in the software modules may put the system under risks. But the prediction of faulty modules are not always possible at the time of development because software is very complex in nature, so whenever the defects are noticed by any member of the team it should be handled by the experts. Another important thing regarding defect prediction is that the testers can focus on the modules, which seems to be defected. This may reduce the development cost and time to the project at the testing phase and lead to a better release. Customer satisfaction has increased with the end product when there are no defects in the software. During 1990s, when no public data sets were available to

Received: November 8th, 2016. Accepted: September 25th, 2018
DOI: 10.15676/ijeei.2018.10.3.3

validate the proposed models, then the researchers used non-public data sets and claimed that their models had the highest performance in defect prediction than other models. But when these models were applied on the public data sets, then some of the models was evaluated as unsuccessful in fault prediction. So, benchmark data sets are very crucial for defect prediction models under dissimilar situations for their validation. In 2005, the PROMISE repository [18] published a number of public NASA data sets. In this work, CM1, JM1, MC1, MC2, PC3 and PC4 data sets are used for experiments. For defect prediction NASA PROMISE data sets are widely used in the studies.

The above mentioned NASA data sets contained the Halstead [15] and McCabe [17] method-level metrics and also some additional metrics related to projects. All the metrics are not unique, in other means some metrics have a measure with the help of some other metrics, so all the metrics are not important for defect prediction. Preprocessing techniques are also important in the software defect prediction. Before constructing a defect prediction model, the following technique may be applied: feature selection technique [22], [23], [34]. With the help of these preprocessing techniques defect prediction performance improved. That's why in this paper, authors' uses different feature selection techniques for finding the most relevant features of the data sets rather than all the features. To the best of our information, this is the first work to find the most relevant features by using many feature selection techniques for software defect prediction. The features are weighted by following feature ranking techniques: Chi-squared statistic, SVM, PCA, Gini index, Information Gain Ratio, correlation, and Symmetrical Uncertain Attributes Evaluation (SUAE). With the help of these feature selection techniques, the rank is provided for each feature of the data sets and the feature extraction is started with the top ranked four features of the data sets and goes up to the twenty three features according to rank bases. Different classification techniques are used for software defects prediction. In this work, Naïve Bays, LibSvm, Multi-Layer Perceptron, Lazy Kstar, Classification via Regression, JRip, J48 and Decision Tree classification techniques are used to classify the defects in two classes either they are defect prone or defect free. These classification techniques are used to predict the defects in each data set which are given above. This is done with the help of all features and with the help of selected features (from top ranked four features to twenty three features) of the data sets. Several performance parameters are also calculated from 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). These parameters are used for examining the effectiveness of the classification techniques.

The main objective of this work is to improve the efficiency of the classification models with the feature selection techniques. Experiments have been performed with both the test groups (all attributes and selected attributes) and after analyzing the results of all the classifiers it is found that the feature selection models have better capability to predict the defects in the software than all the features of the given data sets. The result is better in several aspects like: accuracy, PD, TNR, F-measure, FNR, and area under curve (AUC).

The rest of the paper is structured as follows: Section 2 deals with the related work in this domain. Section 3 explains the proposed feature selection technique and data mining techniques. Section 4 describes the performance parameters which are utilized to validate the experimental results, the data collection with the description, experimental result and discussion. At last, section 5 describes the summary conclusion of this work.

2. Related Work

To measure the defect, Akiyama [21] has developed a model which was based on the lines of code (LOC) to measure the complexity of a software product. In 1976, McCabe [17] introduced cyclomatic complexity metric to defect prediction, whereas Halstead [15] developed a metrics for defect prediction based on the source code. Here, it is also mentioned that McCabe cyclomatic complexity is used before the development phase while the Halstead

metric is used after development of the source code. These metrics were very popular at that time. During 2000, various limitations existed for defect prediction like: if the source code has been changed then defect prediction is more complicated, historical data issue, need of the defect prediction models in software industries and so on. To overcome the above mentioned problems, several studies have been made. Mockus et al. [25], has proposed a model, called "Just in Time (JIT)" which predicts the software defects dynamically. This model became more popular and a number of authors have been applied this model for defect predictions [24], [26]. Another model, named cross project defect prediction is proposed to handle the issue of historical data especially for designing of the new software systems [27-29]. Several studies have been made to identify the needs of the defect prediction models in software industries [30], [32], these are process and source code metrics [31]. From the last two decades, researchers are trying to focus on the models based on the metrics for defect prediction. Now a days, machine learning and statistical techniques are also reported for defect prediction [33], [35].

For software defects prediction, various classification models are used by researchers such as Decision Trees [5], Naïve Bayes [8], Neural Networks [9], Case-based Reasoning [11]. Generally, Support Vector machines (SVMs) performed better than or reasonable against the other machine learning and statistical models in the context of four PROMISE NASA data sets, stated by Elish et al. [6]. Kunmani et al. [10] stated that the neural network classification model performs better than the other statistical models. Menzies et al. [8] stated that the performance of the Naïve Bays model is better than other models even though it is a simple algorithmic model. Kaur and Pallavi [13] discussed different data mining techniques for defect prediction for example classification, clustering, regression and association. In another study, Quah [11] described the software defect prediction by using neural networks model with genetic training strategy. He also predicts the number of defects in the modules, the number of LOC needs to be changed and time required for correction of the defects. All classification models are based on the fault data which are collected from previously released software projects, known as supervised learning or learning by example in the machine learning environment. But this is not always possible that we have fault data of previous projects, it may be possible that the software which is in developing stage is new in its environment and its functionality, in this case we have very few data or no data for learning, so we need new models and new techniques for defect prediction in the software. This type of classification is known as Semi supervised learning. This approach is used when only a few historical data is available for learning. Unsupervised learning is just opposite of supervised learning, means this technique is used when no historical data available for learning (learning by interpretation).

3. Data Mining and Feature Selection Techniques

This section contains the details of the classification models and feature selection techniques which are used in this work. Data mining is of great importance since 1990s. The main objective of the data mining process is extracting and analyzing the valuable information from large data base systems that help the data holders for making the decision effective [16]. Several data mining techniques are utilized to analyze the quality of the software. This paper includes only classification techniques for software defect prediction such as: Naïve Bays, Lib-SVM, Multi-layer Perceptron, Classification-via-Regression, Decision Tree, J48, K-star, and JRip. Classification models categorize the data sets into predefined classes: Binary classes or multilevel classes. Binary classifiers divide the data into two classes such as either "yes' or "no" and the multilevel classifiers categorize the data into many classes such as "very simple", "simple", "complex" or "very complex". Classification models work with many approaches like: learning and testing, leave one out, k cross folds etc. In this study, the 10-fold cross validation is used for all classification models. Selected classification models which are used in this work are discussed in section 3.2.

A. Feature Selection Approaches

All the attributes which are given in the data set are known as "Features" or "characteristics of the data set. Feature selection is an important technique for extracting the most essential feature of the data set which has hundreds or thousands of features. This problem has occurred in several machine learning tasks such as: prediction, regression, and classification etc. The main objective of feature selection approaches is to find the most appropriate feature or a subset of features from a given data set so that these selected features will improve the effectiveness of a model. The appropriate features are extracted from all features by using some weight functions. This weight function assign a weight value to every feature of the data set and according to the assigned weight value, the features which has highest weight value are selected from data set and the rest of the features which are not crucial are ignored (which has very low weight value). The feature selection techniques have many advantages like: it enhances the performance in several aspects (speed, accuracy and simplicity), it also cuts the dimensionality and noise. According to Table 1, the chi squared test assigned 'hal_content' feature has weight value "1.0" means the 'hal_content' attribute is ranked at first position by chi test, and 'programming time' and 'effort' features have weight value "0" so these features are less important by chi squared statistic test. Description of the features of each data set is given in Table 3.

In this work, seven different feature selection techniques are used to rank each attribute of the data set. After ranking all the features, the sum is calculated for all features according to the weight assigned by the selection techniques and by using value of "total sum" the final ranking is provided to each attribute. The weight values of each feature of JM1 data set with their rank are shown in Table 1. Feature "Volume" has been ranked as "1" and the "essential complexity" has been ranked at "18" by the proposed feature selection model (the same can be verified from Table 1). The same process is followed to provide rank to each feature of the other NASA data sets which is used in this paper.

B. Classification Techniques

B.1. Decision Tree (DT)

Decision Tree is an effective technique in many data mining domains for classification of data. Knab et al. [7] used DT technique to predict the fault density in source code of the software system. DT examines the data and produces a tree and the rules, which help to make a decision in predictions. The output of the decision tree is straightforward for people to understand. DT classifies data by arranging the features in order. Every node in the tree represents a feature and the branch represents the value. Classification starts at the root node and moves to the leaf node for prediction of the class that a particular instance belongs to. The DT is developed in the iterative fashion by dividing data into a separate group. This is done by the algorithm which is used to implement it. Many DT can be constructed from a given data set. In these trees, some trees are more effective than the other decision trees. The optimal tree is chosen for prediction, but sometimes it is impossible to find the optimal tree when the search space is very large. An effective algorithm is required or is been developed which can produce the affordable tree with reasonable accuracy. Basically, the greedy approach is utilized to construct the decision tree and the constructed tree is in top-down recursive fashion. The well-known algorithm for constructing decision trees is C4.5. Due to replication of the problem the decision tree becomes significantly more complex in some aspects and it also has efficiency and scalability problems.

Table 1. Weight calculation of each attributes with seven feature selection technique for JM1 data set and their corresponding rank value

Feature's name of JM1data set	Chi Squared Statistic	SVM	PCA	Gini Index	Gain Ratio	SUAE	Co-relation	Total sum	Rank
ec	0.30	0.01	0.00	0.00	0.00	0.03	0.29	0.63	18
difficulty	0.39	0.07	0.00	0.45	0.31	0.03	0.14	1.41	17
dc	0.15	0.00	0.00	0.47	0.68	0.03	0.09	1.42	16
uni_operator	0.05	0.01	0.00	0.56	0.72	0.03	0.22	1.59	15
branch count	0.31	0.03	0.00	0.44	0.78	0.02	0.06	1.64	14
cc	0.29	0.01	0.00	0.48	0.81	0.04	0.06	1.68	13
prog_tme	0.00	0.14	0.06	0.72	0.72	0.04	0.12	1.80	12
loc exe	0.21	0.04	0.00	0.96	0.75	0.04	0.00	1.99	11
no.of_operators	0.35	0.04	0.00	0.84	0.75	0.04	0.01	2.03	10
loc_total	0.19	0.04	0.00	1.00	0.83	0.03	0.01	2.10	9
length	0.42	0.05	0.00	0.79	0.81	0.04	0.01	2.13	8
no.of_operands	0.50	0.07	0.00	0.75	0.81	0.04	0.03	2.19	7
uni_operand	0.38	0.01	0.00	0.84	1.00	0.03	0.09	2.36	6
hal_content	1.00	0.01	0.00	0.61	0.41	0.04	0.34	2.40	5
level	0.70	0.14	0.00	0.41	0.25	0.04	1.00	2.54	4
effort	0.00	0.14	1.00	0.72	0.72	0.04	0.12	2.74	3
error	0.30	0.78	0.00	0.81	0.86	0.03	0.00	2.78	2
v	0.30	1.00	0.00	0.82	0.86	0.04	0.00	3.03	1

B.2. Neural Networks (NN)

Neural networks consists of numerous interconnected neurons (processing elements). Basically the NN is made up of three things: input layer (which takes the input), hidden layer (perform complex functions), and the output layer (where the output is produced). All neurons in the network are inter-connected through a weighted link. The neurons output depends on the activation functions; these activation functions are either linear or non-linear. Neural network is adaptive in nature. It changes the weights so that the classification error is minimized. Multi-layer perceptron is used in this work for defect prediction. It is multi-layer feed forward network. Zheng [3] introduces a cost sensitive neural network model to determine that the software module is defected or not defected. The architecture of the neural networks models is presented in the Figure. 1. Both binary and multiclass classification problems are resolved through the neural networks.

B.3. Support Vector Machine (SVM)

SVM was developed by Vapnik [2] in 1995 for addressing the problem of pattern recognition. Initially, it was developed only for binary classification but later it extended to solve the multiclass classification problems [19]. This function [f: $S^n \rightarrow \{\pm 1\}$] is required to estimate the training data (M x N dimension) x_i and y_i is the class label, where i =1, 2, 3. . . M,

so that, each pattern is classified correctly. Figure. 2 shows the mapping from input to feature space in a binary classification problem by using SVM and the supports vector is those data samples which are lies on the hyper plane. A hyper plane is determined by the SVM so that all the data samples are divided into separate classes. Although, this is the way how the linear and non-linear separable classification is performed but SVM has become very popular to construct a model for software defect prediction.

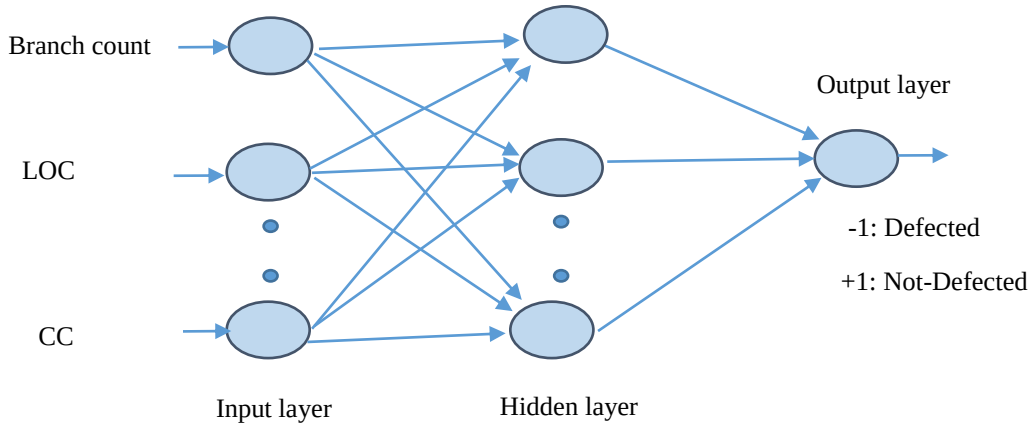

Figure 1. Multi-layer feed forward neural networks for prediction of software faults

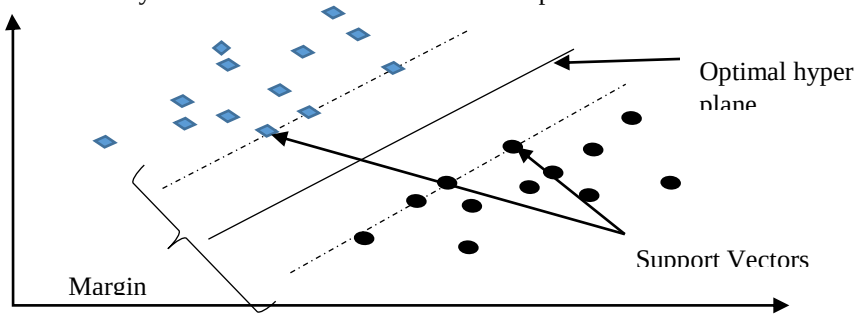

Figure 2. linearly separable classes by support vector machine

B.4. Naïve Bays Classifier

This is the simplest classifier which is relied on Bayesian theorem [20]. This classifier is used for both the feature which is independent to every class and also for those features where independence is no further valid. The performance of naïve bays classifier is quite good even when the features are not independent of each class. This classification technique works in two stages: training stage and prediction stage.

B.5. K-star

K-star [12] classifier is an instance-based classifier. In this classifier, the class of the test instances or cases is relying upon the class of training instances of those are similar to it, which is determined with the help of some similarity function. It uses the distance function which is entropy based.

4. Experiment Description

In this section, details of every data set is given which are used in this work, performance evaluation parameters which are generally used to find the effectiveness of the experimental

results, evaluation of results and the discussion with the outcomes. In this paper, six NASA PROMISE [18] data sets CM1, JM1, MC1, MC2, PC3, and PC4 are used and DT, classification via regression, J48, SVM, MLP, naïve bays, K-star, and J-rip classifiers are used for software defect prediction. Every data set has different number of features and the specified class, that each module is either defective or not defective. All features of the data sets are used with all classification techniques for software defects prediction. Feature selection approaches are used for throwing light on the most reliable features of each data set and these features are used for classification of modules. In this work, seven different techniques are used for feature selection which is discussed in the section 3. By using the result of Table 1, four features are selected for JM1 data set instead of eighteen features. The experiment is also extended from four features to seventeen features. Four to twenty three features are used for defect prediction for the remaining data sets according to their rank values because they have more than thirty features. The accuracy, sensitivity (probability of detection), specificity (true negative rate), precision, probability of false alarm, f-measure, FNR and area under the curve are used as the performance parameters for all the above given techniques by using 10 cross-fold methods. These performance parameters are measured from the confusion matrix, which is generated by the classifiers.

An experiment has three basic needs: data, processing method and the evaluation of the results. In this section, we discuss all the things, which are required for this research work.

Table 2. Features used in this work with their description

S. no.	Name of Attribute	Description	S. no.	Name of Attribute	Description
1	Loc blank	No. of blank lines	19	E	effort
2	Branch count	no. of branches in the code	20	B	effort estimate
3	Call pairs	total no. of call pairs	21	T	programming time
4	Loc code & comment	total LOC with comments	22	Loc exe	executable LOC
5	Loc comment	no. of comment lines	23	decision count	
6	Condition count	no. of condition in code	24	decision density	
7	v(g)	Cyclomatic complexity	25	edge count	
8	iv(g)	McCabe design complexity	26	design density	
9	ev (g)	McCabe essential complexity	27	maintenance severity	
10	µ1	no. of unique operators	28	global data density	
11	µ2	no. of unique operands	29	global data complexity	
12	N1	total operators	30	modified condition count	
13	N2	total operands	31	node count	
14	N	total no. of operands & operators	32	normalized cyclomatic complexity	
15	V	program volume	33	multiple condition count	
16	L	program length	34	parameter count	
17	D	difficulty measure	35	percent comments	
18	I	intelligence	36	loc total	

A. Data sets Characteristics

This section describes the data and their characteristics. The description of all the concerned data sets is given in the Table 2. Every data set has a different type and different number of features, and the details of those features are provided in Table 3.

Table 3. Data sets description

Data set	Number of Modules	Defects (%)
CM1	327	13%
JM1	9371	18%
MC1	264	11%
MC2	127	35%
PC3	1125	13%
PC4	1399	13%

B. Proposed Model

In this paper, most essential features are extracted from the data set according to the assigned weight value, which is provided by the different feature selection approaches as described in section 3. Seven approaches are used to give a rank to each parameter of the data sets and only four to twenty three features are used to classify the data according to defected or not defected. Figure. 3 describes the process of the proposed model.

Table 4. 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)

C. Parameters for Performance Assessment

This section contains the performance assessment parameters which are used in this work to evaluate the performance of all classification techniques, which are discussed in the preceding sections.

Confusion matrix helps to measure the performance parameters, which are generated by the classifiers in the form of classifier's class and the actual class of the module. Table 4 displays the structure of the confusion matrix. Defects prediction has been done with machine learning approach. Training data is given to the classification model with the class label as defected and not defected. Classification models learned from the training data set and then the models classify the modules as either defected or not defected. Generally, classification models produce four types of values which have been shown in the confusion matrix in Table 4. Defected modules are treated as "positive" and not defected modules treated as "negative". Each module of the data set will be residing in one of the four defined field in the confusion matrix. Each field in the confusion matrix has its own meaning as:

- ✓ True positive (TP): how many positive cases are predicted correctly by the classifier.
- ✓ True negative (TN): how many negative cases are predicted correctly by the classifier.
- ✓ False positive (FP): the modules which are not defected but predicted as defected.
- ✓ False negative (FN): the modules which are defected but predicted as not defected.

Confusion matrix is an opportunity to measure the several performance parameters. Some of the parameters which are used in this work are described below:

a) Accuracy (ACC): modules which are correctly identified by the classification technique. TP and TN are the values of the confusion matrix which are predicted correctly. Accuracy is calculated as follows:
   Accuracy (ACC) = (TP + TN) / (TP + TN + FP +FN)
b) Sensitivity or Probability of Detection (PD) or True Positive Rate (TPR): this is the proportion of defected modules that are correctly predicted by the classifier. This is calculated as:
   Sensitivity = (TP) / (TP + FN)
c) Specificity or True Negative Rate (TNR): this is the opposite of TPR, it is the proportion of not defected modules which are correctly classified by the classifier, calculated as:
   TNR = (TN) / (TN + FP)

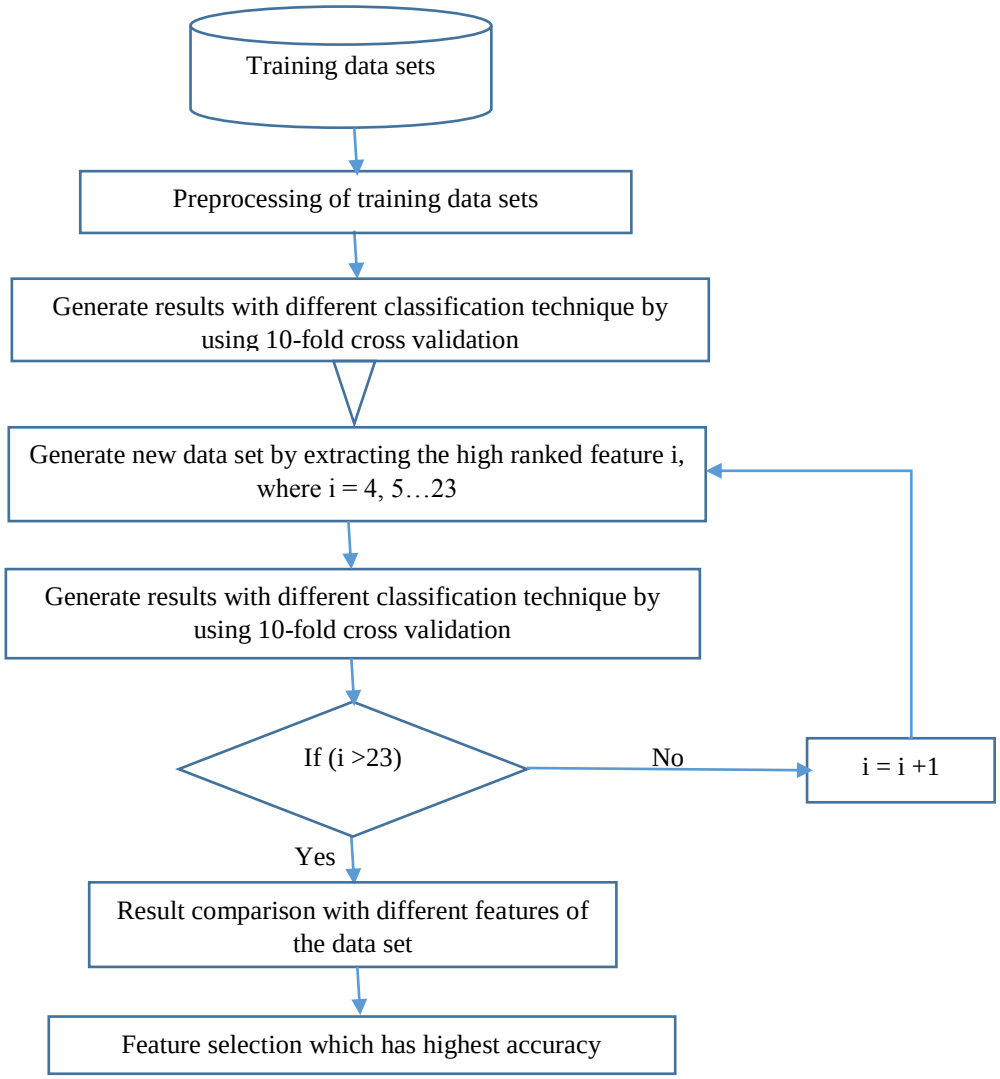

Figure 3. Working of the proposed model

d) Probability of false alarm (PF) or False Positive Rate (FPR): this indicates that the proportion of negative or not defected cases which are predicted as positive or defected. This is measured as:
FPR = (FP) / (FP + TN)
e) Precision: proportion of positive or defected cases which are correct, the following equation is used to measure precision:
Precision = (TP) / (TP + FP)
f) False Negative Rate (FNR): proportion of defected cases which are incorrectly classified as not defected or negative, calculated as: FNR = (FN) / (FN + TP)
g) F-measure: harmonic mean of the sensitivity and precision are used to compute f-measure, this is measured with the help of following equation: F-measure = (2 * Precision * Sensitivity) / (Sensitivity + Precision)
h) Area under Receiver Operating Characteristics Curve (AUC): this is a very important parameter used to measure the performance of different classifiers. It plots the false positive rate on the x-axis and true positive rate on the y-axis. In the initial stage, this was used only for signal detection theory; generally AUC is used for benchmarking.

D. Experimental Results and Discussion

Table 5. Accuracy rate of both the experiments of each data sets

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	87.2	82.1	90.5	72.4	88.4	88.3
4	87.5	81.9	90.9	74.8	87.6	89.2
5	87.5	81.8	90.9	74.8	87.6	89.2
6	87.5	82.0	90.5	76.4	88.4	89.1
7	87.2	82.0	90.5	75.6	88.4	89.0
8	87.2	82.0	91.7	74.0	88.4	89.1
9	87.2	82.1	91.7	74.0	88.4	89.0
10	87.8	82.1	90.2	74.0	88.4	89.3
11	88.1	82.2	90.5	74.8	88.4	89.3
12	88.1	82.2	90.5	74.8	88.4	89.6
13	88.4	82.2	90.5	74.8	88.4	89.4
14	88.4	82.2	90.2	74.8	88.4	89.8
15	88.4	82.2	90.9	74.8	88.4	89.6
16	88.4	78.6	91.3	74.8	88.4	89.4
17	87.2	82.1	91.3	74.8	88.4	89.4
18	87.2	---	91.3	74.8	88.5	90.0
19	87.2	---	90.9	74.8	88.4	89.5
20	87.2	---	90.9	74.8	88.4	89.8
21	87.2	---	90.2	74.8	88.4	89.7
22	87.2	---	91.3	74.8	88.4	89.6
23	87.2	---	91.3	74.8	88.4	89.6
Max	88.4	82.2	91.7	76.4	88.5	90.0

In this section results of NASA PROMISE data sets (shown in Table 2) are analyzed with the help of some performance parameters which is described in the previous section. In this work, two experiment has been done: in the first experiment, all features of the concerned data sets are used with the different classifiers for prediction of the software defects, and in the

second experiment, the most important features are extracted from the data sets (from four features to twenty three features are concerned in this work) by using feature selection techniques for software defect prediction. Classifiers which are outlined in section 3.2 are used for defect prediction. These classifiers generate a confusion matrix of predicted class with the actual class of each module of the data set. Based on the generated confusion matrix, the numerous performance parameters are measured for all the data sets. Accuracy, sensitivity, specificity, precision, f-measure, false negative rate, and area under the curve are measured for performance evaluation. The result of the data sets with their performance parameters is shown in Table 5 to Table10, and from Figure 4 to Figure. 9. In this work, the experiments were finished with 10 fold cross validation. In Table 5, the best measured accuracy of the classifiers with different data sets has been shown for all the features and for the selected feature models. The green color value indicates the highest accuracy rate of a particular data set. Result of Table 5 show that the features selection models have the highest accuracy rate for all data sets than all attributes model.

Sensitivity or probability of detection has been demonstrated in Table 6. This table includes the proportion of positive modules which are also predicted as positive. By analyzing the results, it is found that the feature selection model has more capability for defect prediction than all features, with five of six data sets.

Table 6. PD / Sensitivity of both the experiments of each data set

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	26.2	99.6	55.6	43.2	90.0	99.5
4	21.4	99.7	51.9	38.6	16.4	99.8
5	23.8	99.7	51.9	38.6	16.4	99.8
6	26.2	99.1	51.9	40.9	80.7	99.7
7	26.2	99.1	51.9	38.6	80.7	99.6
8	28.6	99.1	51.9	38.6	78.6	99.6
9	31.0	99.6	51.9	36.4	76.4	99.6
10	28.6	99.6	51.9	38.6	80.0	99.5
11	31.0	99.7	51.9	40.9	80.0	99.5
12	31.0	99.7	51.9	40.9	77.1	99.5
13	31.0	99.6	51.9	47.7	79.3	99.5
14	31.0	99.7	51.9	38.6	80.7	99.6
15	28.6	99.7	51.9	38.6	85.0	99.6
16	28.6	99.7	51.9	40.9	85.0	99.6
17	31.0	99.7	51.9	38.6	87.9	99.6
18	31.0	---	51.9	43.2	87.9	99.6
19	28.6	---	51.9	47.7	86.4	99.6
20	28.6	---	51.9	47.7	87.9	99.8
21	26.2	---	51.9	43.2	90.7	99.6
22	26.2	---	51.9	43.2	89.3	99.6
23	26.2	---	51.9	43.2	90.7	99.5
Max	31.0	99.7	55.6	47.7	90.7	99.8

Table 7 includes the results of specificity. This table contains the proportion of negative modules which are predicted correctly as negative. After analysis of the result, it shows that the performance of feature selection models is better with four out of six data sets and one data set result is same as with all attributes.

Table 7. Precision/specificity of both the experiments for each data set

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	50.0	85.0	62.5	71.4	85.7	90.9
4	57.1	83.2	66.7	78.9	50.0	90.4
5	66.7	83.4	66.7	78.9	50.0	90.4
6	66.7	83.5	62.5	81.8	85.7	91.8
7	50.0	83.6	62.5	73.9	85.7	90.4
8	50.0	83.6	72.7	76.2	85.7	90.5
9	50.0	84.0	72.7	76.2	85.7	90.4
10	60.0	84.0	55.6	76.2	85.7	90.5
11	80.0	84.3	60.0	77.3	85.7	90.8
12	80.0	84.3	60.0	77.3	85.7	90.8
13	66.7	84.6	56.3	77.3	85.7	90.8
14	66.7	84.7	53.3	77.3	85.7	91.2
15	66.7	84.8	66.7	77.3	85.7	91.1
16	66.7	85.0	62.5	77.3	85.7	90.9
17	33.3	84.8	61.1	77.3	85.7	91.3
18	44.4	---	61.1	77.3	85.7	91.2
19	33.3	---	58.8	77.3	85.7	90.9
20	33.3	---	58.8	77.3	85.7	91.0
21	50.0	---	53.3	77.3	85.7	91.0
22	40.0	---	64.3	77.3	85.7	90.9
23	40.0	---	64.3	77.3	85.7	91.1
Max	80.0	85.0	72.7	81.8	85.7	91.8

Table 8 shows the best results of true negative rate of all the classifiers with all features and selected features of the data sets, a major variation can be noted in data set PC4, and the results of remaining data sets of both the models are almost similar.

F-measure is very essential performance parameter for measuring the performance of the classifiers. Table 9 contains the results of f-measure. It is easy to verify from Table 9 that the selected features models performs better with four data sets and the results of remaining two data sets are the same for both models.

The consequence of the FNR is shown in Table 10. In the previous performance parameters, the maximum value considered as the desired value but in case of FNR, minimum value is desirable and this performance parameter can impose major effects on the quality of the software. Because the modules is defected in nature but the classifiers predicted them as not defected, this wrong prediction can be very dangerous in the latter part of the project. So, the classifier should have the capability to predict the defected modules as defected. After analysis of the results of Table 10, it is found that the proposed features selection models have higher probability to predict the positive modules as positive modules or defected modules. The proposed model performed better for all the concerned data sets.

Table 8. TNR of both the experiments for each data set

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	100.0	30.4	100.0	100.0	99.8	37.6
4	100.0	15.8	100.0	100.0	100.0	34.3
5	100.0	17.6	100.0	100.0	100.0	34.3
6	100.0	18.5	100.0	100.0	99.9	46.6
7	100.0	19.2	100.0	92.8	99.9	32.0
8	100.0	19.8	100.0	100.0	99.9	32.6
9	100.0	22.7	100.0	100.0	99.8	31.5
10	100.0	23.2	100.0	100.0	99.8	33.1
11	100.0	25.1	100.0	100.0	99.8	34.3
12	100.0	25.3	100.0	100.0	99.8	37.6
13	100.0	26.9	100.0	100.0	99.8	37.6
14	100.0	27.9	100.0	100.0	99.8	39.9
15	100.0	28.6	100.0	100.0	99.8	38.8
16	100.0	29.7	100.0	100.0	99.8	38.8
17	100.0	29.4	100.0	100.0	99.8	39.3
18	100.0	---	100.0	100.0	99.8	37.6
19	100.0	---	100.0	100.0	99.8	36.5
20	100.0	---	100.0	100.0	99.8	37.1
21	100.0	---	100.0	100.0	99.8	36.5
22	100.0	---	100.0	100.0	99.8	33.7
23	100.0	---	100.0	100.0	99.8	34.3
Max	100.0	30.4	100.0	100.0	100.0	46.6

E. Discussion

The result of different performance parameters of the concerned data sets has been shown in the Table 5, 6, 7, 8, 9, 10 and in Figure. 4, 5, 6, 7, 8, and 9. These tables and Figureures containes the results of both models (all features and selected features). In Table 5, the accuracy is shown, after analysis of the results of Table 5, the authors found that the features selection models provides better defect prediction performance for CM1, JM1, MC1, MC2, PC3, and PC4 data sets (same can be verified from the Table 5), shown in green color. The better result of the Table 6, 7, 8, and 9 is also indicated in green color, features selection models again provide better performance with several aspects like: PD, specitivity, TNR, f-meaure, and FNR. Area under the curve has been depicted in Figure. 4 to Figure.9. Each Figureure contains two graphs. The left side graph shows the AUC with all features of a particular data set and the right side graph shows the AUC of features selection models (proposed model). In the given graph, x-axis indicates the false positive rate and the true positive rate plotted at the y-axis. The Figure. 4 shows the AUC graph of both the experiments for CM1 data set, the value of the left side graph is 0.6839 and 0.7307 for the right side graph. The result suggests that the proposed model which have greater AUC value means have better performance. The AUC plot of JM1 data set have shown in Figure. 5. The values are 0.7089 and 0.7136 for the left and right side respectively. For the MC1 data set the AUC values are 0.7243 on left side graph and 0.7663 for the right side graph. For the PC4 data set the AUC values are 0.8491 and 0.8797 for left and right respectively. And the rest of the Figureures also

indicates the greater AUC value in the right side graph, which shows that the feature selection models have higher AUC values than all feature model for all the data set CM1, JM1, MC1, MC2, PC3 and PC4. When we look at the overall results of all the performance parameters for both the models, then we can conclude that the proposed model provides better defect prediction capability.

Table 9. F-measure of both the experiments for each data set

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	0.25	0.90	0.38	0.49	0.35	0.93
4	0.25	0.90	0.44	0.52	0.24	0.94
5	0.27	0.90	0.44	0.52	0.24	0.94
6	0.28	0.90	0.47	0.55	0.33	0.94
7	0.27	0.90	0.43	0.52	0.34	0.94
8	0.30	0.90	0.45	0.51	0.35	0.94
9	0.31	0.90	0.44	0.49	0.34	0.94
10	0.29	0.90	0.41	0.49	0.35	0.94
11	0.30	0.90	0.41	0.52	0.34	0.94
12	0.29	0.90	0.40	0.52	0.32	0.94
13	0.30	0.90	0.45	0.53	0.32	0.94
14	0.30	0.90	0.40	0.52	0.31	0.94
15	0.30	0.90	0.40	0.52	0.30	0.94
16	0.30	0.90	0.47	0.52	0.30	0.94
17	0.30	0.90	0.49	0.52	0.29	0.94
18	0.30	---	0.49	0.52	0.28	0.94
19	0.28	---	0.45	0.52	0.29	0.94
20	0.26	---	0.38	0.52	0.30	0.94
21	0.26	---	0.45	0.52	0.30	0.94
22	0.26	---	0.44	0.52	0.30	0.94
23	0.52	---	0.44	0.52	0.32	0.94
Max	0.52	0.90	0.49	0.55	0.35	0.94

Table 10. FNR of both the experiments for each data set

Features	CM1	JM1	MC1	MC2	PC3	PC4
ALL	73.8	0.4	44.5	56.8	10.0	0.5
4	78.6	0.3	48.1	61.4	83.6	0.2
5	76.2	0.3	48.1	61.4	83.6	0.2
6	73.8	0.9	48.1	59.1	19.3	0.3
7	73.8	0.9	48.1	61.4	61.4	0.4
8	71.4	0.9	48.1	61.4	19.3	0.4
9	69.0	0.4	48.1	63.6	21.4	0.4
10	71.4	0.4	48.1	61.4	23.6	0.5
11	69.0	0.3	48.1	59.1	20.0	0.5
12	69.0	0.3	48.1	59.1	20.0	0.5

Features	CM1	JM1	MC1	MC2	PC3	PC4
13	69.0	0.4	48.1	52.3	22.9	0.5
14	69.0	0.3	48.1	61.4	20.7	0.4
15	71.4	0.3	48.1	61.4	19.3	0.4
16	71.4	0.3	48.1	59.1	15.0	15.0
17	69.0	0.3	48.1	61.4	15.0	0.4
18	69.0	---	44.1	56.8	12.1	0.4
19	71.4	---	48.1	52.3	12.1	0.4
20	71.4	---	44.1	52.3	13.6	0.2
21	73.8	---	48.1	56.8	12.1	0.4
22	73.8	---	44.1	56.8	9.3	0.4
23	73.8	---		56.8	10.7	0.5
Min	69.0	0.3	44.1	52.3	9.3	0.2

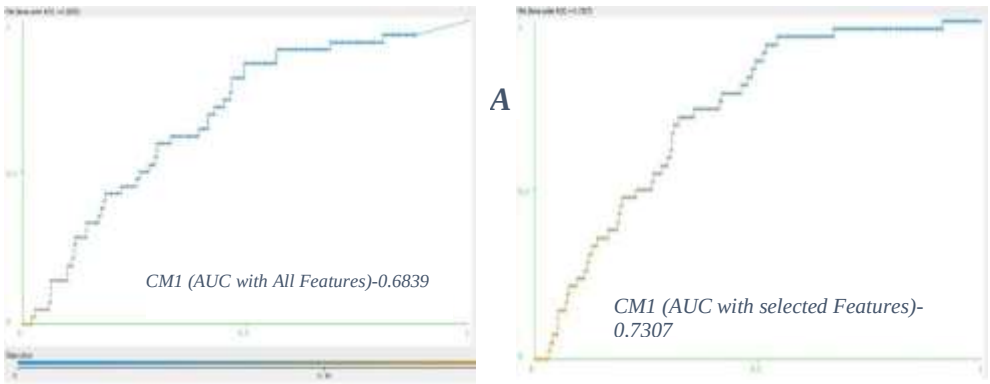

Figure 4. A plot of AUC for CM1 data set of both the experiments

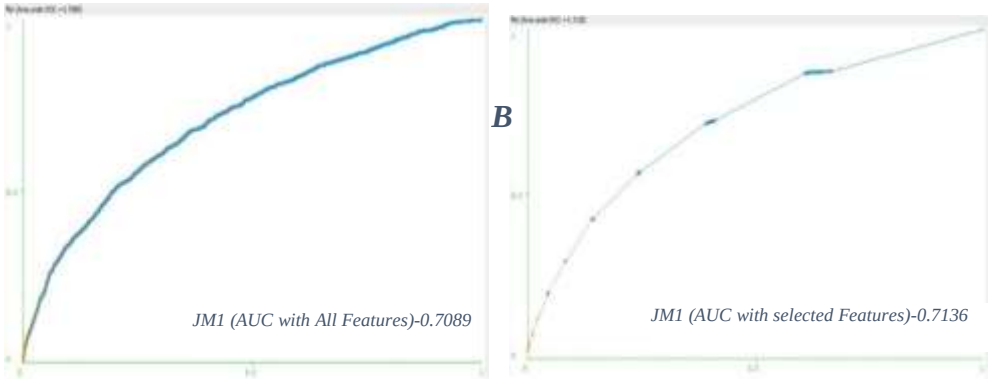

Figure 5. A plot of AUC for JM1 data set of both the experiments

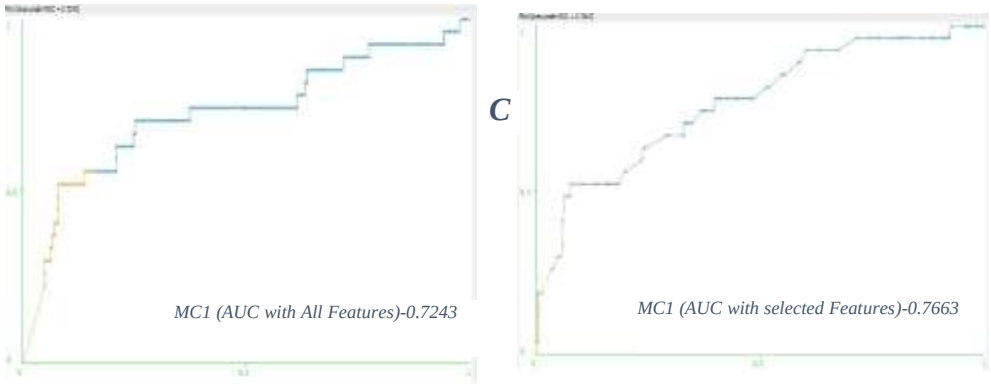

Figure 6. A plot of AUC for MC1 data set of both the experiments

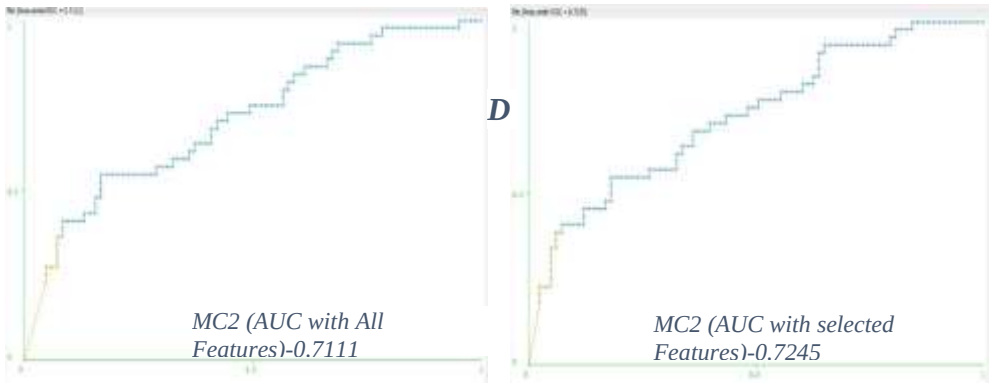

Figure 7. A plot of AUC for MC2 data set of both the experiments

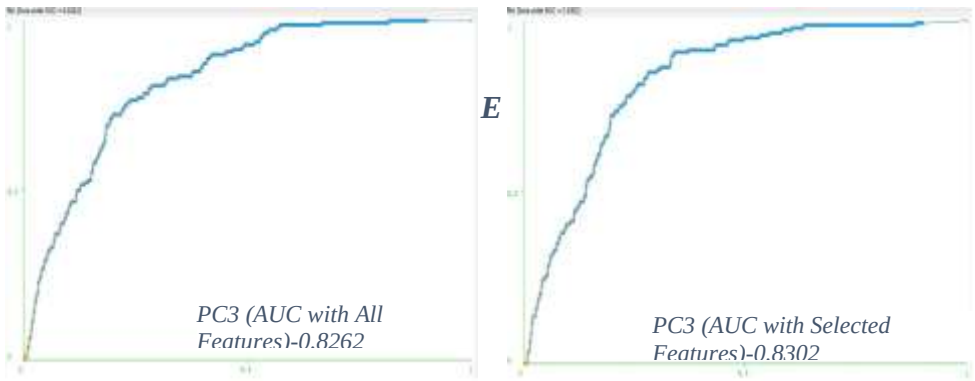

Figure 8. A plot of AUC for PC3 data set of both the experiments

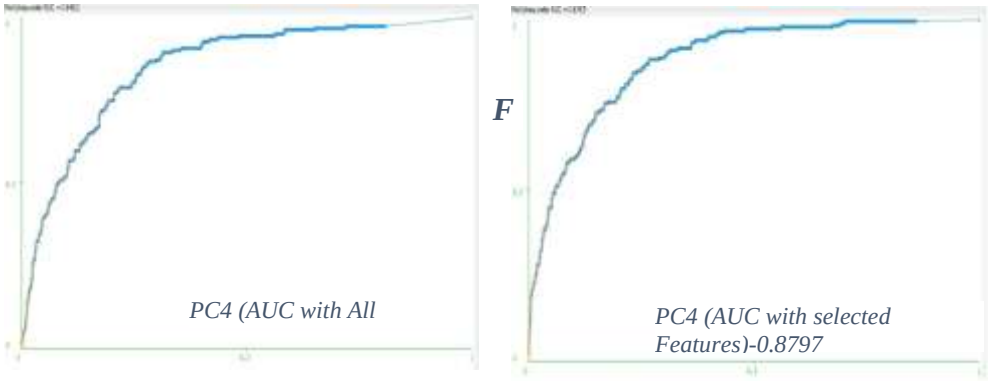

Figure.9: A plot of AUC for PC4 data set of both the experiments

5. Summary and Conclusions

In this study, the authors proposed a feature selection model for software defect prediction. Public NASA data sets CM1, MC1, MC2, PC3, and PC4 from PROMISE data repository, are used. The different classification models are used for defect prediction, and two test groups are performed on each data set. In the first test group, all the features of each data sets are used with the classifier models for fault prediction, but in the second test group, several feature selection techniques are used for assigning weight to every feature of the data set, then only high weighted features are used for software defect prediction. For this work, only four features to twenty three features are used for the second test group experiment. The result of both the test group was calculated with the help of a confusion matrix which has been generated by each classifier and both the test groups were analyzed on the basis of the several performance parameters. The measured results of each table indicated the best value of the actual data set with all the classifiers which are concerned in this work. After analyzing the result of several performance parameters of both the test groups, it is found that, the proposed model performed better in several phases like: accuracy, PD, TNR, FNR, f-measure and AUC and it can be concluded that the proposed test group model have better ability for software fault prediction.

6. References

[1]. Koru, A. Gunes, and Hongfang Liu. "Building effective defect-prediction models in practice." *Software, IEEE*, vol. 22, no. 6, 2005, pp. 23-29.

[2]. Vapnik, Vladimir, "The nature of statistical learning theory", Springer, 2000.

[3]. J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction", *Expert Systems with Applications*, vol. 37, no. 6, 2010, pp. 4537-4543.

[4]. Khoshgoftaar, T. M., Allen, E. B., Ross, F. D., Munik oti, R., Goel, N. & Nandi, A., "Predicting Fault-Prone Modules with Case-Based Reasoning", in *Proc. the Eighth International Symposium on Software Engineering*, 1997, pp. 27.

[5]. M. Evett, T. Khoshgoftaar, P. Chien, E. Allen, "GP-based software quality prediction", in *Proceedings of the Third Annual Genetic Programming Conference, San Francisco, CA*, 1998, pp. 60–65.

[6]. K.O. Elish, M.O. Elish, "Predicting defect-prone software modules using support vector machines", *Journal of Systems and Software*, vol. 81, no. 6, 2008, pp. 649–660.

[7]. Knab, Patrick, Martin Pinzger, and Abraham Bernstein. "Predicting defect densities in source code files with decision tree learners." In Proceedings of the 2006 *International Workshop on Mining Software Repositories*, 2006, pp. 119-125. ACM.

[8]. T. Menzies, J. Greenwald, A. Frank, "Data mining static code attributes to learn defect predictors", *IEEE Transactions on Software Engineering*, vol. 33, no. 1, 2007, pp. 2–13.

[9]. M.M. Thwin, T. Quah, "Application of neural networks for software quality prediction using object-oriented metrics", in *Proceedings of the 19th International Conference on Software Maintenance, Amsterdam, The Netherlands*, 2003, pp. 113–122.
[10]. S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, P. Thambidurai, "Object-oriented software fault prediction using neural networks", *Information and Software Technology*, vol. 49, no. 5, 2007, pp. 483–492.
[11]. Quah, Tong-Seng. "Estimating software readiness using predictive models." *Information Sciences*, vol. 179, no. 4 2009, pp. 430-445.
[12]. John, G. Cleary and Leonard, E. Trigg,"K*: An Instance- based Learner Using an Entropic Distance Measure", *Proceedings of the 12th International Conference on Machine learning*, 1995, pp. 108-114.
[13]. J. Kaur and Pallavi, "Data Mining Techniques for Software Defect Prediction", *International Journal of Software and Web Sciences (IJSWS)*, 2013, pp. 54-57.
[14]. Fenton, Norman E., and Shari Lawrence Pfleeger, "Software metrics: a rigorous and practical approach", *PWS Publishing Co*., 1998.
[15]. M. Halstead, "Elements of Software Science", Elsevier, New York, 1977.
[16]. D. Hand, H. Mannila and P. Smyth, "Principles of data mining", MIT, 2001.
[17]. T. McCabe, "A complexity measure", *IEEE Transactions on Software Engineering*, vol. 2, no. 4, 1976, pp. 308–320.
[18]. S.J. Sayyad, T.J. Menzies, "The PROMISE Repository of Software Engineering Databases", University of Ottawa, Canada, 2005, <http://promise.site.uottawa.ca/SERepository>.
[19]. T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes", *Journal of Artificial Intelligence Research*, vol. 2, 1995, pp. 263–286.
[20]. Langley P, Iba W, Thompson K., "An analysis of Bayesian classifiers", in *Proceedings of the 10th National Conference on Artificial Intelligence*, 1992, pp. 223-228.
[21]. F. Akiyama. An Example of Software System Debugging. In Proceedings of the *International Federation of Information Processing Societies Congress*, pages 353–359, 1971.
[22]. J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In Proceedings of the 2013 *International Conference on Software Engineering, ICSE* '13, pages 382–391, Piscataway, NJ, USA, 2013. IEEE Press.
[23]. S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. In *Proceeding of the 33rd international conference on Software engineering, ICSE* '11, pages 481–490, New York, NY, USA, 2011. ACM.
[24]. T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi. An empirical study of just-in-time defect prediction using cross-project models. In *Proceedings of the 11th Working Conference on Mining Software Repositories, MSR* 2014, pp. 172–181, New York, NY, USA, 2014. ACM.
[25]. A. Mockus and L. G. Votta. Identifying reasons for software changes using historic databases. In *Proceedings of the International Conference on Software Maintenance*, 2000.
[26]. Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A large-scale empirical study of just-in-time quality assurance. *IEEE Trans. Softw. Eng*., 39(6):757–773, June 2013.
[27]. J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In Proceedings of the 2013 *International Conference on Software Engineering, ICSE* '13, pages 382–391, Piscataway, NJ, USA, 2013. IEEE Press.
[28]. B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano. On the relative value of cross company and within-company data for defect prediction. Empirical Software. Eng., 14:540–578, October 2009.

[29]. S. Watanabe, H. Kaiya, and K. Kaijiri. Adapting a fault prediction model to allow inter languagereuse. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering, PROMISE '08*, pages 19–24, New York, NY, USA, 2008. ACM.

[30]. C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. W. Jr. Does bug prediction support human developers? Findings from a google case study. In *International Conference on Software Engineering (ICSE)*, 2013.

[31]. F. Rahman and P. Devanbu. How, and why, process metrics are better. In Proceedings of the 2013 *International Conference on Software Engineering, ICSE '13, pages 432–441, Piscataway, NJ, USA, 2013. IEEE Press.*

[32]. F. Rahman and P. Devanbu. Comparing static bug finders and statistical prediction. In Proceedings of the 2014 *International Conference on Software Engineering, ICSE* '14, 2014.

[33]. M. Li, H. Zhang, R. Wu, and Z.-H. Zhou. Sample-based software defect prediction with active and semi-supervised learning. *Automated Software Engineering*, 19(2):201–230, 2012.

[34]. S. Shivaji, E. Whitehead, R. Akella, and S. Kim. Reducing features to improve code change-based bug prediction. Software Engineering, *IEEE Transactions* on, 39(4):552–569, 2013.

[35]. B. Turhan, A. T. Msrl, and A. Bener. Empirical evaluation of the effects of mixed project data on learning defect predictors. *Information and Software Technology*, 55(6):1101 – 1118, 2013.

Amit Kumar Jakhar, He is a student in the Department of Computer Science and Engineering from Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India. He received his M.E. in Computer Science & Engineering from PEC University of Technology, Chandigarh, India in the year of 2010. He received his B.E. (Honours) from MDU, Rohtak, Haryana, India in the year of 2008. His research interest area is software engineering.

Kumar Rajnish, He is an Assistant Professor in the Department of Computer Science and Engineering from Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India. He received his PhD in Engineering from BIT Mesra, Ranchi, Jharkhand, India in the year of 2009. He received his MCA Degree from MMM Engineering College, Gorakhpur, State of Uttar Pradesh, India. He received his B.Sc. Mathematics (Honours) from Ranchi College Ranchi, India in the year 1998. He has 33 International and National Research Publications. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming languages, Software Estimation, and Cognitive Approach in Software Engg.

Copyright of International Journal on Electrical Engineering & Informatics is the property of School of Electrical Engineering & Informatics, Bandung Institute of Technology and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.