International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.5 (2016), pp.183-198
http://dx.doi.org/10.14257/ijmue.2016.11.5.17

Finding Best Mining Scheme for Development of Multinomial Software Fault Prediction Model

Dipti Kumari and Kumar Rajnish

Birla Institute of Technology, Mesra Ranchi, Jharkhand-835215(INDIA)
Kumari_dipti0511@yahoo.co.in,krajnish@bitmesra.ac.in

Abstract

This paper discuss different classification methods toward reliability and quality improvement of software systems by predicting fault-prone module before testing. Classification capability of Data mining techniques and Object-oriented property based knowledge stored in Object-Oriented metrics are used to classify the software module as fault-prone in different error categories or not fault-prone. Three versions of Eclipse, the java-based Open source Integrated Development environment as dataset for training and testing all the classification based data mining techniques are used. First of all, Threshold base feature ranking (i.e. Area under the ROC curve) is used for selecting effective OO-metrics in building prediction model. After that using those subsets of selected attributes, classification models are built with 41 different classifiers for multinomial classification in fault detection systems. Finally, the performance of a classifier is evaluated with respect to the PRC performance metric. Based on the performance results appropriate classifiers (Random Committee, Random Tree, Randomizable Filtered classifier and IBK) which depict a higher performance and accuracy compared to the others are selected. Our results indicate that Random Tree, Random Committee and Randomizable Filtered Classifier have same performance. IBK classifier also has same performance but little bit less and Kstar has less performance compared to previous four selected classifiers.

Keywords*: OO-metric, classifier, Data mining, ROC, PRC, TBFR, Random Tree, MCC*

1. Introduction

Assuring high quality software is perceived as a key factor to succeed in the software industry. However, deliveries of products of poor quality are still common due to time constraints and limited resources. One way to address this problem is to try to make the software testing process more efficient, and hence find more faults in less time or with fewer resources. For example, fault proneness prediction models can be built on the basis of historic data about changes and faults combined with measures of the structural properties of the software. Assuming that a sufficiently accurate model can be built on the basis of the available data, the model can be applied on a forthcoming release of the software system – giving predictions that identify those software modules that are likely to contain different fault category. Having identified the modules fault category, the testing activities handled by software engineer depending upon their debugging skill capability can focus on those modules category to improve testing efficiency. A significant research effort has been dedicated to defining specific quality measures and building quality models based on those measures. Such models can then be used to help decision-making during development of software systems. Fault-proneness or the numbers of defects detected in a software component (usually a module, class, or file) are the most frequently investigated dependent variables [1]. In this case, we want to predict the fault-proneness of classes in order to potentially finding more defects for the same amount of effort. For example, assuming a class is predicted as “high” to be faulty, Software engineer having high rank in debugging skill would take corrective action by

ISSN: 1975-0080 IJMUE

investing additional effort to inspect and test this class [3]. Given that software development companies might spend between 50 to 80 percent of their software development effort on testing [2], research on fault-proneness prediction models can be motivated by its high cost-saving potential.

To build fault-proneness prediction models there are a large number of modeling techniques to choose from, including standard statistical techniques and data mining techniques [4]. The data mining techniques are especially useful since we have little theory to work with and we want to explore many potential factors (and their interactions) and compare many alternative models so as to optimize effort, cost and time effectiveness.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive overview of related works, whereas Section 3 presents our study design. In Section 4 we report our results, comparing several modeling techniques and sets of measures using a number of different evaluation criteria. Section 5 discusses what we consider the most important threats to validity, whereas Section 6 concludes and outlines directions for future research.

2. Related Work

Data mining is a process to convert raw data into useful information. It is a process designed to explore and analyze large amounts of data to find consistent patterns, trends or relationships among variables and then to validate the findings by applying the observed patterns to new set of data. Data Mining can be divided into two tasks: Predictive tasks and descriptive tasks. The ultimate aim of data mining is prediction and predictive data mining is the most common type of data mining and has the most direct applications to business [28]. Predictive Data mining relies on formulas that compare pass successes and failures, and then uses those formulas to predict future outcomes.

In literature, we found that many research has been done for predicting fault in the Software component. Most of the research for this problem done through statistical techniques, Machine Learning and data mining [5-10]. All the prediction is done on binary classification. But, we have tried to do classification in multinomial aspects. The reason behind this idea is that if we will able to know the module error category before testing then we can allocate for debugging modules to software engineers according to their debugging skill. Debugging skill depends on two terms *i.e*. experience and familiarity with product family [4].

In [6], researchers present a methodology for predicting software faults based on random forest, which is an extension of decision tree learning. Random forest technique was applied in five case studies based on NASA data set. The predictive accuracy of this technique was found to generally higher than that of achieved logistic regression, discriminant analysis and the algorithms in two machine learning software Packages WEKA. Authors in [7] have compared the performance of prediction models by using static attributes of embedded software. Three machine learning algorithms *i.e*. J48, OneR and Naive Bayes have been used for prediction purpose of two datasets. It was found that J48 and OneR performed better than Naïve Bayes learner.

Authors in [5] have applied Support Vector Machines for predicting fault prone software modules and its prediction performance is compared against eight statistical and machine learning methods in the context of four NASA datasets. The results indicate that prediction performance of SVM is generally better than the compared models

In [8], defect prediction is performed using method level metrics and class level metrics and concludes that SVM Method outperforms other classifiers for class level metrics and Random forest shows better performance for method level metrics

In [9], researchers proved that classifier ensembles can effectively improve classification performance than single classifier.

A detailed analysis of some of the existing machine learning techniques for defect prediction has been carried out and that the selection of best learning techniques depends on data insight at that point in time [10].

3. Research Background

In this section, we present the Dataset description (Section 3.1), summary of metrics studied in this article (Section 3.2), collection of error data and categorization (Section 3.3), and empirical data collection (Section 3.4).

3.1. Dataset Description

We have used 3 version of Eclipse (*i.e*. Eclipse2.0, Eclipse2.1 and Eclipse3.0) [13]. The dataset description is given in Table 1.

Table 1. Dataset Description

Version	NFP	Nominal	Low	Mid	High	Total no .of fault prone classes	Total no. of classes
Eclipse2.0	4086	2405	24	121	12	2562	6648
Eclipse2.1	5695	1942	126	6	28	2102	7797
Eclipse3.0	7626	2761	19	89	09	2878	10504

3.2. Software Metrics Studied

The selection of software metrics was a difficult task because there are many available metrics. We used two criteria in our selection process:

- The set of metrics cover all aspects of OO design.
- We have to be able to collect the metrics by using automated tool.

Finally, we selected 24 metrics which are discussed in this section. These metrics are characterized into coupling, cohesion, inheritance, class complexity and class-size metrics. We used JHAWK [11-12] automated tool metric to collect these metrics from the Eclipse source code [13]. JHAWK compiled the source code and give output as each module name and their set of OO metrics.

Table 2. Metrics Description

Sino.	Metrics	Description
1	NOM	Number of Methods
2	LCOM	Lack of Cohesion of Methods
3	AVCC	Average Cyclomatic Complexity
4	NOS	Total number of java statement
5	UWCS	Unweighted Class size
6	INST	No. of instance variables declared
7	PACK	No. of Packages imported
8	RFC	Total Response For class
9	CBO	Coupling between Objects
10	NLOC	Total lines of code in the class
11	FIN	Fan In(Afferent Coupling)
12	DIT	Depth of Inheritance tree
13	COH	Cohesion
14	LMC	No. of local Methods called
15	LCOM2	Lack of cohesion of Methods2

16	MAXCC	Maximum Cyclomatic Complexity
17	FOUT	Fan Out(Efferent coupling)
18	EXT	No. of External Methods called
19	NSUP	Total no. of super classes
20	TCC	Total Cyclomatic Complexity
21	NSUB	Total no. of sub classes
22	MPC	Message Passing Coupling Value
23	INTR	No. of interfaces implemented
24	CC	Class Complexity

3.3. Collection of Error Data

In the next section, we describe how we collected the error data. From [14] where Eclipse bug data set are freely available, we collected the error data from three official releases of the Eclipse project (versions 2.0, 2.1, and 3.0). Pre release bug data are used for study and multinomial categorization has been done on the pre release error data. In this we divide the error category into 5 classes [15].

- No Error: class containing zero error.
- Nominal: class containing error in the range Min<=error<25Q
- Low :class containing error in the range 25Q<=error<50Q
- Medium: class containing error in the range 50Q<=error<75Q
- High: class containing error in the range 75Q<=error<Max

Table 3. Descriptive Statistics of Error Data

Version	Min	25Q	50Q	75Q	Max
Eclipse2.0	1	9	18	29	69
Eclipse2.1	1	6	12	18	24
Eclipse3.0	1	9	18	26	43

3.4. Empirical Data Collection

We have taken 3 version of Eclipse (Eclipse2.0, Eclipse2.1 and Eclipse3.0) as a data source. Steps followed for Eclipse2.0, the first dataset preparation are as follows:

Step1: Download the source code of Open Source Software Eclipse2.0.

Step2: Extract from code the entire file having .java extension as a single class.

Step3: Make a table having name Eclipse2.0 having one column 'class name'(which contain the full path of the .java class).

Step4: Extraction of features (*i.e.*-metrics) using jHAWK tool

```
For (classno=1; classno<=maxclassno; classno++)
 {
   Extract 24 features
  }
```

Step5: Add all the features in the table Eclipse2.0 and give the column name according to the feature name.

Step6: From the bug dataset (promise2.0 (a)) of Eclipse2.0 extract pre-release bug and add one more column in the table Eclipse2.0 as pre and insert the value in the column corresponding to the class name.

Step7: Categorization of bug according to value of pre column of Eclipse2.0.

Step8: Find descriptive statistics of pre column, from that we are able to know the min, different number of occurrences of error (nonzero) and max value of error data

Step9: Again find the descriptive statistics of (Min, 25Q, 50Q, 75Q and Max) the different occurrences of number of errors (from min (nonzero) to max).

Step10: Multinomial Categorization

```
If ((pre>1) && (pre<=25Q))
   Category="nominal";
 Else if ((pre>25Q) && (pre<=50Q))
   Category="low";
 Else if ((pre>50Q) && (pre<=75Q))
    Category="mid";
 Else if ((pre>75Q) && (pre<=max))
    Category="high";
 Else
    Category="fault prone";
```

Step11: Add one more column Multinomial insert value in the respective value according to the categorization criteria.

Step12: Do the step1 to step12 for Eclipse2.1 and Eclipse3.0.

This will produce 3 datasets (D) for Eclipse2.0, Eclipse2.1 and Eclipse3.0 respectively as well as complete the dataset preparation process.

4. Research Method

In this section a brief description of Threshold based feature ranking (TBFR) is presented. Moreover, the process of Random Committee and Random Tree classification, which are proved to be the most efficient classifiers amongst the set of classifiers in our research, are enlightened.

4.1. Threshold Based Feature Ranking (TBFR)

In a given set of software metrics, it is likely that some of them are superfluous in characterizing the project's knowledge. Some of them provide redundant knowledge, or provide no new information, or in some cases, have an adverse effect on the defect prediction model. For example, recent studies demonstrate performance improvement of defect prediction models when irrelevant and redundant features are re-moved before modeling [16-18].Feature selection is the process of choosing a subset of features. It is broadly classified as feature ranking and feature subset selection, where feature ranking sorts the attributes according to their individual predictive power, and feature subset selection finds subsets of attributes that collectively have good predictive power. Filters are feature selection algorithms in which a feature subset is selected without involving any learning algorithm. In the context of software defect prediction, we use Threshold-Based Feature Selection technique. It belongs to the filter based feature ranking techniques category. Area under the ROC (Receiver Operating Characteristic) Curve (AUC) TBFR feature ranker is considered in this study [15].

Step1: Dataset D with OO-metrics O_j, j=1, 2... 24

Step2: Each class x belongs to D is assigned to one of 5 classes C(x) belongs to {nfp, nominal, low, mid, high};

Step3: The value of OO-metric O_j for instance x is denoted as $O_j(x)$;

Step4: For j=1… 24 do

Step5: Calculate AUC using attribute O_j and class category at various decision thresholds in the distribution of O_j. The optimal AUC is used.

Step6: Feature ranking is done using optimal average AUC for sorting the OO-metrics according to their individual predictive power.

Step 7: After that, we select the subset of OO-metrics that have good predictive power.

4.2. Classification

Classification is learning a function that maps data items into one of several predefined classes. Examples of classification methods used as part of knowledge discovery applications include classifying trends in financial markets and automated identification of objects of interest in large images databases[20].In fault detection systems, classifiers are used to predict whether each module contains fault in different category and no fault. Several works has been carried out in the field of classification with WEKA tool. We have used this tool for performance analysis of different classification techniques [19].

4.2.1. Random Committee

Random Committee [20] builds an ensemble of weak classifiers and averages their predictions. Random Committee distinguishes itself by making each classifier be based on the same data but using a different random number seed. This only makes sense if the base classifier is randomized; otherwise the classifiers would all be the same.

4.2.2. Random Tree

The Random Trees classifier takes the input feature vector, classifies it with every tree in the forest, and outputs the class label that received the majority of “votes”. This results in the classification rules and generates the Misclassification Rate, Precision and Recall performance measures [20-21].

4.2.3 Randomizable Filtered Classifier (RFC)

Used for running an arbitrary classifier on data that has been passed through an arbitrary filter. Like the classifier, the structure of the filter is based exclusively on the training data and test instances will be processed by the filter without changing their structure [24].

4.2.4 IBk

It uses the instances themselves from the training set to represent what are learned, and be kept. When an unseen instance is provided the memory is searched for the training instance.

4.2.4 K*

It is an instance-based learning algorithm that stores all training instances and does not build a model until a new instance need to be classified and they use some domain specific distance function to retrieve single most similar instance from the training set.

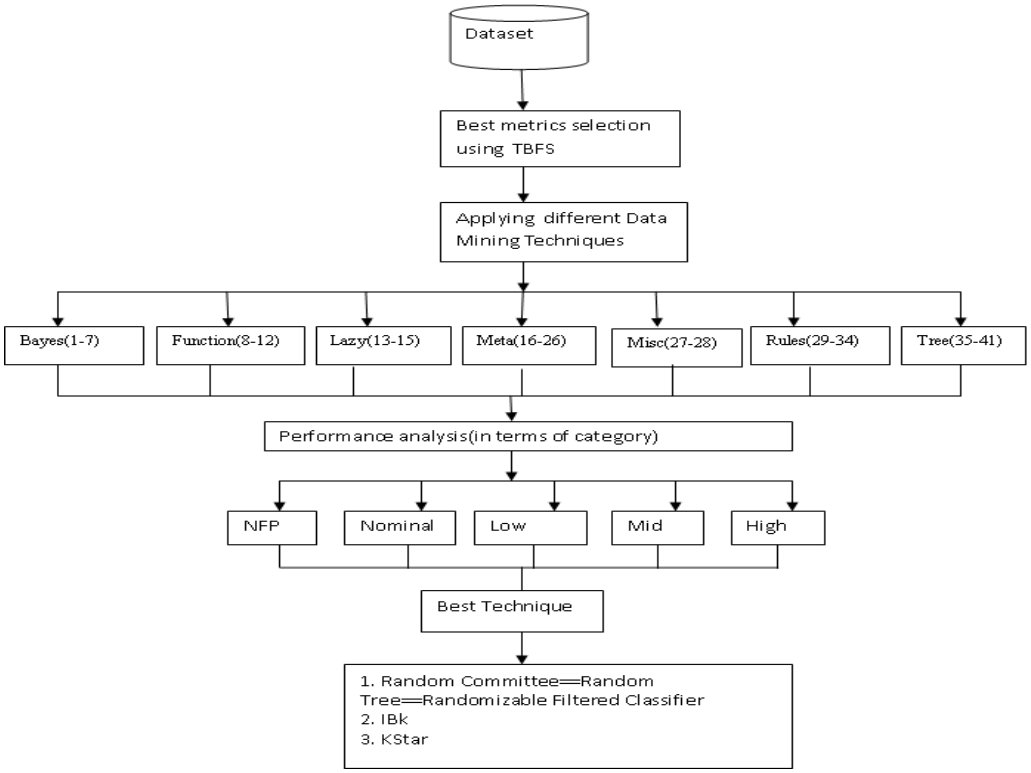

Figure 1. Comparison of Classification Algorithms in Proposed Data Mining Framework

4.3. Performance Measures

Different performance attributes are discussed in this section. These attributes are very helpful to analyze the results of experiments. For defect prediction WEKA tool is used to classify the datasets. The prediction of classification is matched with the actual class of that data. We have done multinomial classification which is depicted in Figure 2.

Actual \ Predicted	F0	F1	F2	F3	F4
F0	T0(true category 0)	F01	F02	F03	F04
F1	F10	T1(true category 1)	F12	F13	F14
F2	F20	F21	T2(true category 2)	F23	F24
F3	F30	F31	F32	T3(true category 3)	F34
F4	F40	F41	F42	F43	T4(true categor 4)

Figure 2. Confusion Matrix for Multinomial Classification

Where TN=T0, TPi =T1, T2, T3 and T4
Where TN=T0, TP_i =T1,T2 ,T3 and T4

$$FN = \sum_{j=1}^{4} F0j \tag{1}$$

Where T0, T1, T2, T3 and T4 are the number of correct predictions that an instance is

$$FP_i = \sum_{i=0}^{4}\sum_{j=1}^{4} F_{ij} \quad \text{Where i} \neq \text{j} \tag{2}$$

negative, positive in nominal category, low category, mid category and high category. Where F_{01} are the number of incorrect predictions that an instance is actually in negative but predicted in nominal category. In F_{ij} first subscript showing the actual instance and second one is showing the predicted result.

The precision indicates that how many defect prone modules are correctly predicted. The precision is measured as in equation 3.

$$PRECISION_i = \frac{TP_i}{TP_i + FP_i} \tag{3}$$

The Recall is equivalent to sensitivity (*i.e.* TPR, hit rate)

$$RECALL_i = \frac{TP_i}{TP_i + FN} \tag{4}$$

F-measure combines precision and recall in the form of harmonic mean of precision and recall:

$$F - measure_i = 2.\frac{PRECISION_i.RECALL_i}{PRECISION_i + RECALL_i} \tag{5}$$

The MCC can be calculated directly from the confusion matrix using the formula:

$$MCC_i = \frac{TP_i \times TN - FP_i \times FN}{\sqrt{(TP_i + FP_i)(TP_i + FN)(TN + FP_i)(TN + FN)}} \tag{6}$$

Precision-Recall Curve (PRC) presents an alternate approach to the visual comparison of classifiers [24]. PR curve can reveal the difference between algorithms which is not apparent from an ROC curve. In a PR curve, x-axis represents recall and y-axis is precision. PR curves favor classifiers which offer high recall and high Precision, *i.e.*, the ideal performance goal lies in the right-hand upper corner. We are not saying that PR curves are better than ROC curves. We only recommend that when ROC curves fail to reveal differences in the performance of different classification algorithms, PR curves may provide adequate distinction. In multinomial categorization and unbalanced data it is also more useful than ROC [25].

5. Experimental Analysis

5.1. TBFR Method for Feature Selection

After ranking the 24 features among those 17 features are selected for developing the model. Subsequently, we used the Sensitivity and Specificity values (they indicate efficiency in classifying faulty Classes) to order the metric values in Table4. We noticed that the size metrics (NOS and UWCS) came before the (CC and RFC) metrics. This information tells that the size metrics are better indicators of faulty classes. These results showed that the threshold values differed from one release to another [15]. With the highest Sensitivity value as the selection standard, we choose the final threshold values for the NOS, UWCS, CC, RFC, NLOC, EXT, MPC, LMC, TCC, PACK, NOM, LCOM2, INST, CBO, MAXCC, FOUT and AVCC metrics and result is summarized in Table 4.

These values can be used by developers as a guideline for designing classes, if the metrics exceed the threshold value then; there are chances of error prone classes.

Table 4. Selected Metrics for Building Model with Rank

metrics	E2.0 thr.	E2.1 thr	E3.0 thr	E2.0 auc	E2.1 auc	E3.0 auc	E2.0 sensitivity	E2.1 Sensitivity	E3.0 Sensitivity	E2.0 1-specificity	E2.1 1-specificity	E3.0 1-specificity	Rank of use
NOM	10.50	12.50	9.83	0.79	0.82	0.77	0.77	0.80	0.75	0.25	0.23	0.31	11
AVCC	1.73	2.16	1.98	0.67	0.71	0.68	0.72	0.74	0.67	0.37	0.31	0.37	17
NOS	61.00	103.83	75.83	0.83	0.86	0.79	0.82	0.80	0.75	0.25	0.20	0.26	1
UWCS	14.83	18.83	16.83	0.81	0.87	0.80	0.77	0.85	0.76	0.26	0.24	0.26	2
INST	3.83	8.17	4.17	0.75	0.81	0.78	0.70	0.79	0.76	0.29	0.25	0.30	13
PACK	9.83	9.83	6.33	0.84	0.83	0.72	0.80	0.79	0.65	0.18	0.24	0.37	10
RFC	34.50	43.50	31.83	0.83	0.85	0.77	0.80	0.81	0.72	0.21	0.20	0.29	4
CBO	3.83	2.83	3.50	0.79	0.76	0.74	0.76	0.77	0.75	0.24	0.42	0.32	14
NLOC	77.83	135.83	92.17	0.82	0.85	0.79	0.82	0.80	0.75	0.27	0.20	0.28	5
LMC	1.50	2.50	1.50	0.82	0.84	0.77	0.83	0.82	0.71	0.24	0.26	0.35	8
LCOM2	21.83	22.83	18.83	0.81	0.80	0.76	0.79	0.77	0.72	0.18	0.26	0.28	12
MAXCC	4.83	5.83	4.83	0.75	0.77	0.73	0.73	0.78	0.70	0.28	0.27	0.34	15
FOUT	1.83	1.83	1.17	0.74	0.79	0.72	0.71	0.76	0.74	0.23	0.33	0.47	16
EXT	23.50	28.50	18.83	0.84	0.85	0.76	0.80	0.81	0.70	0.17	0.19	0.31	6
TCC	23.50	30.50	23.50	0.80	0.82	0.78	0.77	0.79	0.75	0.20	0.21	0.26	9
MPC	20.17	28.50	18.83	0.84	0.85	0.76	0.81	0.81	0.70	0.17	0.20	0.31	7
cc	25.50	38.17	27.50	0.81	0.87	0.79	0.84	0.80	0.74	0.21	0.21	0.28	3

5.2 Results and Discussion

Due to the variety of classifiers the WEKA supports and also its efficient environment for experimental data analysis, we used the WEKA software. Lessman *et al*. [26] and Haghighi *et al*. [4] already had conducted research in this field. According to Lessman opinion classifiers do not differ much from each other. Haghighi extended their research by studying more classifiers in determining an appropriate classifier in fault detection system (*i.e*. detection of class either as a error prone or as a error-free).And finally their results indicated that Bagging classifier has the highest performance in fault-detection .We will extend their research by studying more classifier for our Software fault Prediction problem in Multinomial categorization (*i.e*. prediction of classes in different error categories).Therefore, the values of Precision, Recall, F-measure, MCC, PRC Area, Correctly classified classes and Incorrectly classified classes are calculated for several (*i.e*.41) classifier and eventually according to these values, the best classifier is selected. The Precision, Recall, F-measure, MCC and PRC Area are recommended as the primary accuracy indicators for comparative studies in software fault prediction with imbalanced data in multinomial categorization [25]. Table 5 shows the results of evaluating 41 classifiers on Eclipse2.0 dataset. In order to compare the performance of the mentioned classifiers, MCC, PRC Area, Correctly classified and incorrectly classified criteria are chosen. In the table, the rows of those classifiers which are containing the highest all those value are highlighted with red font color. As it is illustrated in Table 5, Random Committee, Random Tree and Randomizable filtered Classifier usually depict a high MCC, PRC and high correctly classified classes and low incorrectly classified classes' dataset.

Second best classifier is IBk and at the third position KStar is showing best performance. Hence we have chosen Random Committee, Random Tree and Randomizable filtered Classifiers as the appropriate classification algorithm for fault prediction system.

5.3 Random Committee, Random Tree, Randomizable Filtered Classifier, IBk and K-star Performance Evaluation

In this section, Random Committee, Random Tree and Randomizable Filtered Classifier, which were chosen according to our experiments, are compared against the IBk and KStar classifiers that managed to illustrate acceptable results in the Eclipse2.0 dataset. We determined the best approach by evaluating classifiers on Eclipse2.0 which is

the first version of Eclipse. In order to evaluate Random Committee, Random Tree and Randomizable Filtered Classifier as well as their performance on other datasets, we conduct our experiments on other two successive versions of Eclipse2.0 (*i.e*. Eclipse2.1 and Eclipse3.0) datasets. Table 6 and Table 7 illustrate the outcome of comparison. By testing Random Committee, Random tree, Randomizable filtered classifier, IBk and KStar on Eclipse2.1 and Eclipse3.0, Precision, Recall, F-measure, MCC and PRC area were determined. As it is depicted in Table 6 and Table 7. Random Committee, Random tree and Randomizable Filtered Classifier manage to outperform the other two rival approaches, IBk and KStar, in other two successive versions of Eclipse datasets. IBK also shows almost same performance with selected three classifiers.

Mean Absolute Error (M.A.E), Root Mean Square Error (R.M.S.E), Relative Absolute Error (R.A.E) and Root Relative Squared Error (R.R.S.R) are other error related metric value for finding the performance of selected classifiers. The mean absolute error (MAE) is defined as the quantity used to measure how close predictions or forecasts are to the eventual outcomes. The root mean square error (RMSE) is defined as frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. It is a good measure of accuracy, to compare the forecasting errors within a dataset as it is scale-dependent. Relative error is a measure of the uncertainty of measurement compared to the size of the measurement. The root relative squared error is defined as a relative to what it would have been if a simple predictor had been used. More specifically, this predictor is just the average of the actual values.

From the graph Figure1 and Figure2 plotted using Table 8, it is observed that KStar attains highest error rate *i.e*. Kstar has not as much good classification capability as compared to other four. These four classification algorithm are showing same value for all column. It means all four having same classification capability and contains least error rate.

Table 5. Investigating the Performance of Classifiers on Eclipse2.0

Classifier	Class	Precision	Recall	F-measure	MCC	PRC area	Correct Class	Incorrect Class
[1]. A1DE	NFP	0.72	0.791	0.754	0.312	0.797	3234	852
	Nominal	0.547	0.467	0.504	0.257	0.559	1123	1282
	Low	0.4	0.25	0.308	0.314	0.296	6	18
	Mid	0.337	0.231	0.275	0.268	0.238	28	93
	High	1	0.333	0.5	0.577	0.529	4	8
[2]. A2DE	NFP	0.729	0.861	0.79	0.384	0.829	3519	567
	Nominal	0.629	0.465	0.535	0.336	0.644	1118	1287
	Low	0.833	0.208	0.333	0.416	0.471	5	19
	Mid	0.632	0.198	0.302	0.348	0.431	24	97
	High	1	0.167	0.286	0.408	0.663	2	10
[3]. Bayes Net	NFP	0.742	0.627	0.679	0.272	0.776	2560	1526
	Nominal	0.457	0.449	0.453	0.148	0.447	1080	1325
	Low	0.058	0.417	0.102	0.148	0.076	10	14
	Mid	0.072	0.347	0.119	0.125	0.067	42	79
	High	0.074	0.5	0.129	0.189	0.233	6	6
[4]. Naive Bayes	NFP	0.659	0.915	0.766	0.221	0.736	3739	347
	Nominal	0.538	0.164	0.252	0.129	0.48	395	2010
	Low	0.043	0.083	0.056	0.055	0.066	2	22
	Mid	0.085	0.041	0.056	0.047	0.093	5	116
	High	0.051	0.583	0.095	0.169	0.232	7	5
[5]. Naive Bayes Multinomial	NFP	0.639	0.93	0.757	0.145	0.564	3799	287
	Nominal	0.483	0.073	0.126	0.061	0.339	175	2230
	Low	0.019	0.167	0.034	0.046	0.009	4	20
	Mid	0.108	0.083	0.093	0.8	0.029	10	111
	High	0.083	0.25	0.125	0.142	0.027	3	9
[6]. Naive Bayes Multinomial Updateable	NFP	0.639	0.93	0.757	0.145	0.565	3799	287
	Nominal	0.489	0.077	0.132	0.065	0.34	184	2221
	Low	0.016	0.125	0.028	0.035	0.009	3	21
	Mid	0.103	0.083	0.092	0.077	0.03	10	111
	High	0.083	0.25	0.125	0.142	0.027	3	9
[7]. Naive Bayes Updateable	NFP	0.659	0.915	0.766	0.221	0.736	3739	347
	Nominal	0.538	0.164	0.252	0.129	0.48	395	2010
	Low	0.043	0.083	0.056	0.055	0.066	2	22
	Mid	0.085	0.041	0.056	0.047	0.093	5	116
	High	0.051	0.583	0.095	0.169	0.232	7	5
[8]. LibSVM	NFP	0.91	0.998	0.952	0.874	0.91	4078	8
	Nominal	0.995	0.837	0.909	0.872	0.892	2014	391
	Low	1	0.833	0.909	0.913	0.834	20	4
	Mid	1	0.934	0.966	0.966	0.935	113	8
	High	1	0.917	0.957	0.957	0.917	11	1
[9]. Logistic	NFP	0.68	0.903	0.776	0.284	0.79	3688	398
	Nominal	0.584	0.289	0.386	0.216	0.52	694	1711
	Low	0.5	0.083	0.143	0.203	0.259	2	22
	Mid	0.375	0.099	0.157	0.186	0.228	12	109
	High	1	0.167	0.286	0.408	0.29	2	10
[10]. Multilayer Perceptron	NFP	0.673	0.928	0.78	0.284	0.779	3791	295
	Nominal	0.621	0.259	0.365	0.227	0.554	622	1783
	Low	0.667	0.083	0.148	0.235	0.206	2	22
	Mid	0.5	0.058	0.104	0.166	0.206	7	114
	High	0	0	0	0	0.001	0	12
[11]. Simple Logistic	NFP	0.679	0.905	0.776	0.282	0.791	3698	388
	Nominal	0.583	0.284	0.382	0.213	0.519	684	1721
	Low	0	0	0	0	0.252	0	24
	Mid	0.407	0.091	0.149	0.186	0.221	11	110
	High	1	0.167	0.286	0.408	0.235	2	10
[12]. SMO	NFP	0.66	0.942	0.777	0.231	0.658	3851	235
	Nominal	0.574	0.195	0.291	0.165	0.403	469	1936
	Low	0	0	0	0	0.004	0	24
	Mid	0	0	0	0	0.03	0	121
	High	0	0	0	0	0.002	0	12
[13]. Ibk	NFP	0.973	0.998	0.985	0.962	0.999	4077	9
	Nominal	0.996	0.995	0.975	0.962	0.998	2297	108
	Low	1	0.958	0.979	0.979	0.998	23	1
	Mid	1	0.975	0.987	0.987	0.999	118	3
	High	1	1	1	1	1	12	0
[14]. Kstar	NFP	0.964	0.997	0.98	0.949	0.999	4075	11
	Nominal	0.995	0.938	0.966	0.948	0.997	2256	149
	Low	1	0.958	0.979	0.979	0.998	23	1
	Mid	1	0.975	0.987	0.987	0.999	118	3
	High	1	1	1	1	1	12	0
[15]. LWL	NFP	0.637	0.981	0.772	0.193	0.79	4007	79
	Nominal	0.559	0.083	0.145	0.098	0.505	200	2205
	Low	0	0	0	0	0.176	0	24
	Mid	0	0	0	0	0.162	0	121
	High	0	0	0	0	0.159	0	12
[16]. Adaboost	NFP	0.652	0.956	0.776	0.233	0.715	3907	179
	Nominal	0.573	0.157	0.246	0.146	0.457	377	2028
	Low	0	0	0	0	0.005	0	24
	Mid	0	0	0	0	0.026	0	121
	High	0	0	0	0	0.003	0	12
[17]. Attribute Selected classifier	NFP	0.773	0.89	0.827	0.506	0.862	3637	449
	Nominal	0.728	0.565	0.636	0.476	0.745	1358	1047
	Low	0.632	0.5	0.558	0.561	0.491	12	12
	Mid	0.78	0.322	0.456	0.496	0.533	39	82
	High	0.778	0.583	0.667	0.673	0.628	7	5
[18]. Bagging	NFP	0.846	0.954	0.897	0.715	0.941	3899	187
	Nominal	0.861	0.715	0.781	0.681	0.887	1719	686
	Low	0.818	0.375	0.514	0.553	0.571	9	15
	Mid	0.962	0.207	0.34	0.442	0.677	25	96
	High	0.833	0.417	0.556	0.589	0.705	5	7
[19]. Classification Via Regression	NFP	0.702	0.882	0.782	0.331	0.801	3602	484
	Nominal	0.607	0.378	0.466	0.276	0.583	910	1495
	Low	0.5	0.042	0.077	0.144	0.297	1	23
	Mid	0.833	0.124	0.216	0.318	0.441	15	106
	High	1	0.167	0.286	0.408	0.757	2	10
[20]. FilteredClassifier	NFP	0.746	0.894	0.813	0.45	0.813	3653	433
	Nominal	0.685	0.494	0.574	0.399	0.653	1189	1216
	Low	0	0	0	0	0.077	0	24
	Mid	0.818	0.074	0.136	0.244	0.253	9	112
	High	0	0	0	0	0.071	0	12

Classifier	Class	Precision	Recall	F-measure	MCC	PRC area	Correct Class	Incorrect Class
[21]. LogitBoost	NFP	0.677	0.908	0.776	0.278	0.79	3710	376
	Nominal	0.589	0.279	0.378	0.215	0.54	670	1735
	Low	0.75	0.125	0.214	0.305	0.32	3	21
	Mid	0.462	0.099	0.163	0.208	0.22	12	109
	High	1	0.167	0.286	0.408	0.54	2	10
[22]. MultiClass Classifier	NFP	0.672	0.91	0.773	0.262	0.79	3719	367
	Nominal	0.565	0.26	0.356	0.189	0.5	626	1779
	Low	0	0	0	0	0.16	0	24
	Mid	1	0.033	0.064	0.18	0.21	4	117
	High	1	0.083	0.154	0.288	0.2	1	11
[23]. Multiclass Classifier Updateable	NFP	0.673	0.982	0.773	0.199	0.67	4014	72
	Nominal	0.573	0.084	0.146	0.104	0.41	201	2204
	Low	0	0	0	0	0	0	24
	Mid	0	0	0	0	0.03	0	121
	High	0	0	0	0	0	0	12
[24]. Random Committee	NFP	0.976	0.997	0.987	0.965	1	4075	11
	Nominal	0.995	0.96	0.977	0.965	1	2308	97
	Low	1	0.958	0.979	0.979	1	23	1
	Mid	1	0.975	0.987	0.987	1	118	3
	High	1	1	1	1	1	12	0
[25]. Randomizable Filtered Classifier	NFP	0.976	0.997	0.987	0.965	1	4075	11
	Nominal	0.995	0.96	0.977	0.965	1	2308	97
	Low	1	0.958	0.979	0.979	1	23	1
	Mid	1	0.975	0.987	0.987	1	118	3
	High	1	1	1	1	1	12	0
[26]. Random SubSpace	NFP	0.788	0.939	0.857	0.59	0.88	3837	249
	Nominal	0.789	0.584	0.671	0.538	0.8	1404	1001
	Low	0	0	0	0	0.5	0	24
	Mid	0	0	0	0	0.36	0	121
	High	0	0	0	0	0.64	0	12
[27]. CHIRP	NFP	0.833	0.969	0.896	0.711	0.83	3960	126
	Nominal	0.888	0.686	0.774	0.682	0.72	1649	756
	Low	1	0.375	0.545	0.612	0.38	9	15
	Mid	0.957	0.182	0.306	0.414	0.19	22	99
	High	1	0.583	0.737	0.763	0.58	7	5
[28]. FLR	NFP	0.621	1	0.766	0.126	0.62	4086	0
	Nominal	0.738	0.019	0.036	0.075	0.37	45	2360
	Low	1	0.083	0.154	0.288	0.09	2	22
	Mid	1	0.025	0.048	0.156	0.04	3	118
	High	1	0	0	0	0	0	12
[29]. Decision Table	NFP	0.703	0.884	0.783	0.336	0.8	3612	474
	Nominal	0.618	0.387	0.476	0.288	0.57	930	1475
	Low	0	0	0	0	0.06	0	24
	Mid	0.375	0.025	0.047	0.093	0.18	3	118
	High	0	0	0	0	0.02	0	12
[30]. DTNB	NFP	0.796	0.638	0.709	0.368	0.8	2608	1478
	Nominal	0.545	0.545	0.545	0.287	0.55	1310	1095
	Low	0.103	0.417	0.165	0.202	0.22	10	14
	Mid	0.116	0.488	0.188	0.211	0.16	59	62
	High	0.068	0.25	0.016	0.036	0.17	3	9
[31]. FURIA	NFP	0.757	0.6	0.669	0.284	0.7	2405	1636
	Nominal	0.481	0.678	0.563	0.254	0.44	1631	774
	Low	1	0.167	0.286	0.408	0.19	4	20
	Mid	0.824	0.116	0.203	0.305	0.15	14	107
	High	0	0	0	0	0.4	0	12
[32]. Jrip	NFP	0.714	0.854	0.778	0.341	0.71	3490	596
	Nominal	0.595	0.428	0.498	0.288	0.49	1030	1375
	Low	0	0	0	0	0.02	0	24
	Mid	0.781	0.207	0.327	0.397	0.25	25	96
	High	0	0	0	0	0.02	0	12
[33]. OneR	NFP	0.681	0.898	0.774	0.281	0.67	3670	416
	Nominal	0.603	0.316	0.415	0.244	0.44	760	1645
	Low	0	0	0	0	0	0	24
	Mid	0	0	0	0	0.02	0	121
	High	0	0	0	0	0	0	12
[34]. PART	NFP	0.778	0.931	0.848	0.561	0.91	3805	281
	Nominal	0.819	0.566	0.669	0.549	0.81	1361	1044
	Low	0.917	0.458	0.611	0.647	0.66	11	13
	Mid	0.872	0.562	0.683	0.696	0.73	68	53
	High	0.875	0.583	0.7	0.714	0.73	7	5
[35]. J48	NFP	0.84	0.94	0.887	0.687	0.93	3840	246
	Nominal	0.857	0.703	0.772	0.669	0.87	1691	714
	Low	0.765	0.542	0.634	0.643	0.63	13	11
	Mid	0.895	0.562	0.69	0.705	0.77	68	53
	High	0.7	0.583	0.636	0.638	0.6	7	5
[36]. J48 Consolidated	NFP	0.757	0.613	0.677	0.291	0.75	2504	1582
	Nominal	0.473	0.455	0.464	0.169	0.43	1094	1311
	Low	0.102	1	0.185	0.314	0.42	24	0
	Mid	0.14	0.843	0.24	0.32	0.24	102	19
	High	0.194	1	0.324	0.438	0.75	12	0
[37]. J48graft	NFP	0.84	0.94	0.887	0.687	0.93	3840	246
	Nominal	0.875	0.703	0.772	0.669	0.87	1691	714
	Low	0.765	0.542	0.634	0.643	0.63	13	11
	Mid	0.895	0.562	0.69	0.705	0.77	68	53
	High	0.7	0.583	0.636	0.638	0.6	7	5
[38]. LMT	NFP	0.735	0.872	0.797	0.405	0.82	3562	324
	Nominal	0.642	0.47	0.543	0.35	0.63	1131	1274
	Low	0.727	0.333	0.457	0.491	0.45	8	16
	Mid	0.714	0.124	0.211	0.293	0.37	15	106
	High	0.833	0.417	0.556	0.589	0.54	5	7
[39]. Random Forest	NFP	0.956	0.992	0.973	0.93	0.99	4052	34
	Nominal	0.978	0.923	0.95	0.924	0.99	2221	184
	Low	1	0.792	0.884	0.889	0.98	19	5
	Mid	0.991	0.893	0.939	0.939	0.99	108	13
	High	1	0.833	0.909	0.913	0.98	10	2
[40]. Random Tree	NFP	0.976	0.997	0.987	0.965	1	4075	11
	Nominal	0.995	0.96	0.977	0.965	1	2308	97
	Low	1	0.958	0.979	0.979	1	23	1
	Mid	1	0.975	0.987	0.987	1	118	3
	High	1	1	1	1	1	12	0
[41]. REP Tree	NFP	0.789	0.865	0.825	0.516	0.86	3535	551
	Nominal	0.691	0.619	0.653	0.474	0.71	1489	916
	Low	0.429	0.125	0.194	0.23	0.22	3	21
	Mid	0.5	0.025	0.047	0.108	0.21	3	118
	High	0	0	0	0	0.12	0	12

Table 6. Performance Measure of Selected Metrics in Eclipse 2.1

	class	Precision	Recall	F-measure	MCC	PRC area	Correctly classified	Incorrectly Classified
Random Committee	NFP	0.986	0.999	0.993	0.972	1	5691	4
	Nominal	0.998	0.958	0.977	0.97	0.998	1860	82
	Low	1	1	1	1	1	126	0
	Mid	1	1	1	1	1	6	0
	High	1	1	1	1	1	28	0
Random Tree	NFP	0.986	0.999	0.963	0.972	1	5691	4
	Nominal	0.998	0.958	0.977	0.97	1.998	1860	82
	Low	1	1	1	1	1	126	0
	Mid	1	1	1	1	1	6	0
	High	1	1	1	1	1	28	0
Randomizable filtered classifiers	NFP	0.986	0.999	0.993	0.972	1	5691	4
	Nominal	0.998	0.958	0.977	0.97	0.998	1860	82
	Low	1	1	1	1	1	126	0
	Mid	1	1	1	1	1	6	0
	High	1	1	1	1	1	28	0
Ibk	NFP	0.986	0.999	0.993	0.972	1	5691	4
	Nominal	0.998	0.958	0.977	0.97	0.998	1860	82
	Low	1	1	1	1	1	126	0
	Mid	1	1	1	1	1	6	0
	High	1	1	1	1	1	28	0
Kstar	NFP	0.979	1	0.989	0.959	0.999	5693	2
	Nominal	0.999	0.937	0.967	0.957	0.996	1819	123
	Low	1	0.992	0.996	0.996	1	125	1
	Mid	1	1	1	1	1	6	0
	High	1	1	1	1	1	28	0

Table 7. Performance Measure of Selected Classifiers for Eclipse 3.0

	class	Precision	Recall	F-measure	MCC	PRC area	Correctly classified	Incorrectly Classified
Random Committee	NFP	0.982	0.971	0.977	0.969	0.998	7579	47
	Nominal	0.99	0.994	0.992	0.97	1	2682	79
	Low	1	0.947	0.973	0.973	0.997	18	1
	Mid	1	0.989	0.994	0.994	1	88	1
	High	1	1	1	1	1	9	0
Random Tree	NFP	0.982	0.971	0.977	0.969	0.998	7579	47
	Nominal	0.99	0.994	0.992	0.97	1	2682	79
	Low	1	0.947	0.973	0.973	0.997	18	1
	Mid	1	0.989	0.994	0.994	1	88	1
	High	1	1	1	1	1	9	0
Randomizable filtered classifiers	NFP	0.982	0.972	0.977	0.969	0.998	7577	49
	Nominal	0.99	0.994	0.992	0.97	1	2684	77
	Low	1	0.947	0.973	0.973	0.997	18	1
	Mid	1	0.989	0.994	0.994	1	88	1
	High	1	1	1	1	1	9	0
Ibk	NFP	0.982	0.972	0.977	0.969	0.998	7577	49
	Nominal	0.99	0.994	0.992	0.97	1	2684	77
	Low	1	0.947	0.973	0.973	0.997	18	1
	Mid	1	0.989	0.994	0.994	1	88	1
	High	1	1	1	1	1	9	0
Kstar	NFP	0.996	0.937	0.966	0.955	0.996	7617	9
	Nominal	0.978	0.999	0.988	0.956	0.999	2587	174
	Low	1	0.947	0.973	0.973	0.997	18	1
	Mid	1	0.989	0.994	0.994	1	88	1
	High	1	1	1	1	1	9	0

Table 8. Error Rate of Selected Classifiers in Eclipse2.1 and Eclipse3.0

Algorithm	MAE	RMSE	RAE	RRSE
Random Committteee	0.61	5.53	3.79	19.46
Random Tree	0.61	5.53	3.79	19.46
Randomizable Filtered Classifier	0.61	5.53	3.79	19.46
Ibk	0.63	5.53	3.89	19.46
Kstar	1.11	6.83	6.85	24.03

Algorithm	MAE	RMSE	RAE	RRSE
Random Committee	0.68	5.82	4.21	20.5
Random Tree	0.68	5.83	4.2	20.5
Randomizable Filtered Classifier	0.69	5.82	4.27	20.5
Ibk	0.69	5.82	4.27	20.5
Kstar	1.26	7.29	7.81	25.6

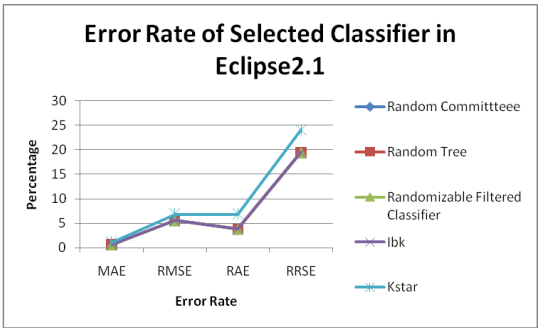

Figure 1. Error Rate of Selected Classifiers

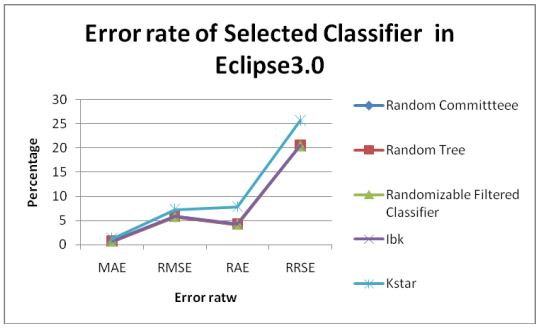

Figure 2. Error Rate of Selected Classifiers

6. Conclusion

TBFR method ranked all metrics based on their discrimination ability using ROC curve and we found that 17 metrics are best predictors among 24 metrics. Data mining help us in the extraction of useful knowledge from large data repositories. In this paper, we have done comparison of 41 classification algorithms over Eclipse first version Eclipse2.0 dataset. By comparing all 41 classification algorithms we figured that Random tree, Random committee, Randomizable Filtered Classifier show same result.IBK also has same performance result compared to selected three random natured classifiers. These four classifiers also show a better performance than the rest of classifiers for fault prediction system. KStar classifier is also showing performance somewhat less than those four classifiers but better than rest. So, we can choose any classifier among those four classifiers as appropriate classifiers. For verification of selected classifiers, they were compared on two successive versions of Eclipse. The results illustrated that all classifies which are random in nature have highest performance on fault prediction. Therefore by employing those four classifiers, the prediction system is more accurate

As future work, we can focus on these four random nature algorithm and through its optimization for fault detection systems, increase the detecting performance.

References

[1] Taylor, "Applications of data mining in Software Engineering", International Journal of Data Analysis Techniques and Strategies, vol. 2, no. 3, **(2010)**, pp. 243-257.
[2] A. V. K. Prasad and Dr. S. R. Krishn1, "Data Mining for Secure Software Engineering-Source Code Management Tool Case Study", International Journal of Engineering Science and Technology, vol. 2, no. 7, **(2010)**, pp. 2667-2677.
[3] K. Dipti and R. Kumar, "Investigating the Effect of Object-oriented Metrics on Fault Proneness Using Empirical Analysis", International Journal of Software Engineering and Its Applications, vol. 9, no. 2, **(2015)**, pp. 171-188.
[4] A. A. Shahrjooj and M. A. Haghighi, "Appling Mining Schemes to Software Fault Prediction: A Proposed Approach Aimed at Test Cost Reduction", Proceedings of the World Congress on Engineering, **(2012)**.
[5] K. O. Elish and M. O. ELish, "Predicting defect prone software modules using Support Vector Machines", Journal of Systems and Software, vol. 81, no. 5, **(2008)**, pp. 649-660.
[6] L. Guo, Y. Ma, B. Cukic and H. Singh, "Robust Prediction of fault proneness by Random Forest", Software Reliability Engineering, , ISSRE, **(2004)**.
[7] P. Singh, "Comparing the effectiveness of machine learning algorithms for defect prediction", International Journal of Information Technology and Knowledge management, vol. 2, no. 2, **(2009)**, pp. 481-483.
[8] A. Shanthini and R. M. Chandrasekaran, "Applying machine learning for fault Prediction using Software Metrics", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 6, **(2012)**.
[9] T. Wang, W. Li, H. Shi and Z, Liu, "Software Defect Prediction on Classifier Ensemble", Journal of Information & Computational Sciences, vol. 8, no. 16, **(2011)**, pp. 4241-4254.
[10] V. U. B. Challagulla, F. B. Bastani, I. L. Yen and R. A. Paul, "Empirical Assessment of Machine Learning based Software defect Prediction Techniques", Proceedings of the 10th IEEE International Workshop on Object Oreinted Real time Dependable Systems, **(2005)**.
[11] JHAWK metrics reference http://www.virtualmachinery.com/jhawkreferences.html, **(2013)**.
[12] JHAWK metrics reference http://www.aivosto.com/project/help/pm-oomisc.html, **(2013)**.
[13] Eclipse source code (for archived releases): http://archive.eclipse.org/eclipse/downloads/, **(2012)**.
[14] Eclipse bug data (for archived releases): http://www.st.cs.uni-sb.de/softevo/bug-data/eclipse, **(2012)**.
[15] K. Dipti and R. Kumar, "Finding error-prone classes at design time using class based Object-Oriented metrics threshold through statistical method", INFOCOMP, Journal of Computer science, vol. 12, no. 1, **(2013)**, pp. 49-63.
[16] K. Gao, T. M. Khoshgoftaar and H. Wang, "An empirical investigation of filter attribute selection techniques for software quality classification", In Proceedings of the 10th IEEE International Conference on Information Reuse and Integration, **(2009)**, pp. 272–277.
[17] H. Wang, T. M. Khoshgoftaar and J. V. Hulse, "A comparative study of threshold-based feature selection techniques", In IEEE International Conference on Granular Computing, **(2010)**, pp. 499-504.
[18] H. Wang, T. M. Khoshoftaar and N. Seliya, "How Many Software Metrics Should be Selected for Defect Prediction", In Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference, **(2011)**, pp. 69-74.
[19] K. Yugal and G. Sahoo, "Analysis of Parametric & Non Parametric Classifiers for Classification Technique using WEKA", International Journal of Information Technology and Computer Science (IJITCS), vol. 4, no. 7, **(2012)**, pp. 43.
[20] T. B. Hu, "Introduction to Knowledge Discovery and Data Mining", Institute of Information Technology, National Center for Natural Science and Technology.
[21] I. H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, **(2005)**.
[22] K. Dipti and R. Kumar, "Comparing efficiency of software Fault Prediction Models developed through binary and Multinomial Logistic Regression Techniques", Second International conference on Information System Design and Intelligent Applications(INDIA-2015), Proceeding published in the Springer's series on Advances in Intelligent Systems and Computing, vol. 340, **(2015)**, pp. 187-197.
[23] S. G. Jacob and Dr. R. G. Ramani, "Discovery of Knowledge Patterns in Clinical Data through Data Mining Algorithms: Multi-classCategorization of Breast Tissue Data", International Journal of Computer Applications (IJCA), October DOI: 10.5120/3920-5521. Published by Foundation of Computer Science, New York, vol. 32, no. 7, **(2011)**, pp. 46-53.
[24] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves", In ICML '06: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ACM, **(2006)**, pp. 233-240.

[25] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets", PLoS One, e0118432, vol. 10, no. 3, **(2015)**.
[26] S. Lessmann, B. Baesens, C. Mues and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings", IEEE Trans. on Software Engineering, vol. 34, no. 4, **(2008)**, pp. 485-496.

Authors

Dipti Kumari, Completed M. Tech. (CSE) from Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India in the year 2010. Currently, she is pursuing PhD on Software Fault Prediction. Her Research area is Object-Oriented Metrics, Software Engineering, Software Fault Prediction, Programming Languages, Database Management System and Object-Oriented Software fault prediction.

Kumar Rajnish, is an Assistant Professor in the Department of Information Technology at Birla Institute of Technology, Mesra, Ranchi, Jharkahnd, 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 23 Research Publications. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming Languages, and Database System.