Determining Bug Severity using Machine Learning Techniques

K. K. Chaturvedi and V. B. Singh

Abstract — *Software Bug reporting is an integral part of software development process. Once the Bug is reported on Bug Tracking System, their attributes are analyzed and subsequently assigned to various fixers for their resolution. During the last two decades Machine-Learning Techniques (MLT) has been used to create self-improving software. Supervised machine learning technique is widely used for prediction of patterns in various applications but, we have found very few for software repositories. Bug severity, an attribute of a software bug report is the degree of impact that a defect has on the development or operation of a component or system. Bug severity can be classified into different levels based on their impact on the system. In this paper, an attempt has been made to demonstrate the applicability of machine learning algorithms namely Naïve Bayes, k-Nearest Neighbor, Naïve Bayes Multinomial, Support Vector Machine, J48 and RIPPER in determining the class of bug severity of bug report data of NASA from PROMISE repository. The applicability of algorithm in determining the various levels of bug severity for bug repositories has been validated using various performance measures by applying 5-fold cross validation*[1].

Index Terms —Machine Learning, Supervised Classification, Feature Selection, Bug Severity.

I. Introduction

Bug reporting and fixing is an important phase of software development, refinement and maintenance for both open and close source projects. Various bug reporting systems have been developed for submission or reporting of a bug and tracking their progress of fix. Bug reporting and tracking systems provide a platform to record the problems/failures faced by the client or user of the software [16]. It helps in fixing the next release as well as measuring the quality of software by applying different mathematical and statistical models. It is very important that the critical bug need to be fixed on priority basis. The urgency of the bug requires a parameter that needs to be provided during bug reporting process. The criticality of the bug is specified by the severity of the bug that also needs to be filled up during the bug reporting process. Bug severity is the degree of impact that a defect has on the development or operation of a component or system. Bug severity can be classified into different levels based on its impact on the system. Bug severity has different number of levels based on the application of the software. NASA defines five severity levels of the bug reports from the fullest to the dullest or from 1 to 5 in terms of the numeric value. The value 1 represents the most severe bug and has major impact on the system while value 5 represents cosmetic changes or messages/labels on the screen that used to appear during the operation of the system. The severity is an important attribute that will be helpful in finding the urgency of its fix. Severity of the fault has been also classified based on debugging time lag [30].

The summary and description of the reported bug contains running summarized and detailed textual information about the bug. This textual information of the already submitted bug reports having defined severity levels is quite useful in determining the severity level, priority and assignment to the prospect fixer for the newly submitted bug report using supervised classification. This can be analyzed by applying one of the sub areas of data mining, i.e. text mining. Text mining can be utilized to mine the knowledge from the textual description [8, 10, 26, and 27]. Machine learning techniques have been used in literature to determine the duplicate bug report [11, 12, and 29], severity [3, 4, and 5], assignee for its fix [13, 14, 15, 20, and 21] and bug prioritization [19] but a complete study on the applicability of machine learning technique in determining the bug severity has been missing in the literature.

Text mining has been applied in determining the change requests as a new feature or enhancement [2]. A study has been conducted in assigning the severity level from 1 to 3 based on important, non-important and request for enhancement of the bug report [6]. In the available literature, various efforts have been made in determining the severity of a reported bug on the basis of their textual description. An attempt has been made on textual description of bug reports of closed source NASA's PITS (Project and Issue Tracking System) data using RIPPER, a rule based learner to predict the severity level of the newly submitted bug report [3, and 24]. Another attempt has been made to predict severity levels based on the description of the newly submitted bug report in binary classes, i.e. severe and non-severe using the Naive Bayes classification [4]. This study has been extended to compare with a few other data mining algorithms such as Naive Bayes Multinomial, K-Nearest Neighbor and Support Vector Machine for classification of bug severity into binary classes for open source projects [5]. To the best of our knowledge,

[1]K.K. Chaturvedi is research scholar at Department of Computer Science, University of Delhi, Delhi, and scientist at Indian Agricultural Statistics Research Institute, Library Avenue, PUSA, New Delhi, India (e-mail: kkcchaturvedi@gmail.com).

V. B. Singh is with Delhi College of Arts & Commerce, University of Delhi, Delhi, India (e-mail: vbsinghdcacdu@gmail.com).

very few efforts have been made in determining various class levels of bug severity (1 to 5) by applying these machine learning techniques on summary attribute of the bug reports. Summary attribute of the bug report will provide very useful and concise information about the bug. In the paper, an attempt has been made to demonstrate the applicability of popularly known and matured machine learning techniques in determining various class levels of severity of the newly reported bug based on multi-class classification. We have evaluated the performance of developed models by applying 5-fold cross validation and calculated various performance measures namely, precision, recall, F-measure and accuracy to present the applicability of machine learning in determining the bug severity.

The paper is organized into six sections. Section II describes the materials and methods required to perform the analysis. Section III deals the classifier building and evaluation. Section IV describes the results and discussions on applicability of machine learning techniques. Finally, the paper is concluded in section V with future research directions in section VI.

II. MATERIALS AND METHODS

The bug reports data for the NASA's PITS projects are made available in the PROMISE (Predictor Models in Software Engineering) repositories [1]. The bug repositories have been downloaded from the repositories. The number of bug reports in each project is shown in the table I.

TABLE I. SEVERITY WISE NUMBER OF REPORTED BUGS AND NUMBER OF REDUCED FEATURE (TERM) SETS

Projects	Severity wise Number of reported bugs						Number of Terms
	1	*2*	*3*	*4*	*5*	*Total*	
PitsA	0	111	155	34	3	303	697
PitsB	0	23	523	382	59	987	1001
PitsC	0	0	132	177	7	316	309
PitsD	0	1	167	13	1	182	347
PitsE	0	24	517	243	41	825	1059
PitsF	0	9	477	209	48	743	999

These data sets have been downloaded as comma separated values (csv) files, the reports are transformed into text files using a program written in java and a standard process of preprocessing these bug reports have been applied. This process starts with tokenization, stop word removal, word stemming using Porter stemming algorithm [7], and user defined list of stop words.

Terms are also filtered out having length less than 3 and more than 50 characters. The remaining number of terms for each considered data set is shown in table 1. The information gain is an entropy based measure and feature selection has been done using information gain criteria. In this way top 25, 50, 75, 100, 125, 150, 175 and 200 terms are selected to build the models of different datasets. The bug reports are represented using matrix where each row represents the reported bugs while each column represents the stem of the terms/feature and last column contains the severity level of the reported bug. The cells of the matrix are filled using TF*IDF measure which is a combination of term frequency (TF) and inverse document frequency (IDF). TF is the number of occurrence of a term in the particular document and IDF is logarithm of ratio of total number of documents and number of documents where the term appears.

Text mining is a combination of pre-processing of textual data and application of data mining algorithms. In general, text mining is a specialized data mining process to identify the trends when the data contains texts. Document classification is widely studied in Machine Learning [9 and 26]. Classification or categorization is the way of automatically assigning a document to a predefined category. In other words, it is defined as categorizing the documents based on their topic of relevance. Classification is broadly categorized either supervised or unsupervised. In supervised classification, the classes of the training data set are already known whereas in unsupervised classification, the classes of the data are unknown. Various predictive methods for analyzing unstructured information have been summarized using text mining [27]. In the current study, classification algorithms Naive Bayes, k-Nearest Neighbor, Naive Bayes Multinomial, Support Vector Machine, J48 (Java implementation of C4.5) and Repeated Incremental Pruning to Produce Error Reduction (RIPPER) have been used to classify the severity level of the newly submitted bug report based on the previously submitted bug reports.

A. Naïve Bayes Classifier(NB)

A naïve Bayes classifier estimates the class-conditional probability with assumption that all attributes are conditionally independent. In this technique, the conditional probabilities for each term with the given class will be calculated. After calculating the conditional probabilities of each term with respect to every class, a new report can be classified based on the sum of the probabilities for each class of each term occurring within the document [22].

B. k-Nearest Neighbor (k-NN)

The k-nearest neighbor compares every report in the dataset with the given report and find out the similarity with the existing report. The similarity can be obtained by using any similarity or distance measure. The data point is classified based on the class labels of its neighbors. In case, data points have more than one label, the new report will be assigned to the majority of the vote class. In this technique, k is the number of neighboring points. For example, if k=3, the nearest three points will be considered and find out the majority of the class. If there is a tie, then the report may be assigned to any of tied class [10].

C. Naïve Bayes Multinomial (NBM)

This is similar to the naïve Bayes classifier except that the terms also consider its weight during the probability calculation. This will capture the length of the document and

the occurrences of the terms in number of documents. This can be handled by using TF*IDF measure.

D. Support Vector Machine(SVM)

This is a mathematical and statistical learning technique and has shown promising results empirically in many applications started with handwriting recognition techniques to bioinformatics application for identification of genes [17]. This technique has also been applied in text mining because it works well with the high dimensions of data and avoids the curse of dimensionality problem. The algorithm tries to separate the report set into the given categories based on the maximum margin hyper plane [28]. The distance between these hyper planes will be maximized for separating report set into binary classes. If the data can not be separated using linear hyper plane, the features can be transformed using suitable kernel function. The choice of kernel function is depend on the type of problems and data set. The SVM technique has also been used for feature reduction in the literature. Multi-class classification is handled in SVM using one against one, one against many and directed acyclic graph (DAG) SVM [23 and 25].

E. J48 (Java Implementation of C4.5)

J48 is an open source Java implementation of the C4.5 algorithm in the WEKA data mining tool and also available for RapidMiner as WEKA extension. C4.5 Algorithm [32], an extension of Quinlan's ID3 (Interactive Dichotomizer 3) algorithm [31] is developed by Ross Quinlan and is used to generate a decision tree which is being used for classification.

F. (Repeated Incremental Pruning to Produce Error Reduction (RIPPER)

RIPPER, is a covering algorithm that runs over the data in multiple passes and also called as rule covering algorithms [24]. This algorithm learns one rule at each pass for the majority class. W-JRIP is an open source Java implementation of WEKA data mining tool which is available in RapidMiner as WEKA extension.

Learned model can be validated using n-fold cross validation by using various performance measures namely, precision (the percentage of correctly predicted documents), recall (the percentage of predicted out of the total number of documents), f-measure (twice of the harmonic mean of precision and recall) and accuracy (correctly predicted documents for all classes).

III. Classifier Building and Evaluation

Many methods/techniques of machine learning have been applied for classification of bug severity level. These techniques can also be applied on various phases of knowledge discovery process which starts from data pre-processing to knowledge representation. The techniques used are Naive Bayes, Naive Bayes Multinomial, k-Nearest Neighbor (1-NN, 2-NN, 3-NN and 4-NN using cosine similarity), Support vector machine with three degree polynomial kernel function, J48 and RIPPER.

The models have been built using operators available in RapidMiner [18] by varying number of terms i.e., 25 to 200 in interval of 25 based on the information gain criteria. These terms are quite helpful in building the model and can be used to predict the severity levels for the upcoming bug report.

The learned classifiers have been evaluated by 5-fold cross validation approach using stratified sampling for splitting the reports between training and testing set. In each case, a classification model was trained using 80% of the data and tested on the remaining 20% to compute mean precision, mean recall, and the mean F-measures. For each data set, the experiments have been repeated on these trained models 5 times, rotating the 20% testing part of the data. We have validated the applicability of machine learning techniques for all PITS projects (PitsA to PitsF). The performance measure namely, accuracy and F-Measure for all NASA projects have been calculated. The accuracy has also been obtained for the learned classifier models as shown in Table II. The graphs have been plotted for accuracy as shown in figure 1. The F-measure have been plotted and shown in figure 2 with respect to the number of terms considered based on various machine learning techniques for PitsA project. The maximum and minimum values of F-Measures for various predicted severity levels have also been plotted in the figure 3 to figure 8 for the project PitsA to PitsF respectively.

IV. Results and Discussions

Accuracy of various machine learning techniques for PITS projects are shown in Table II. From table II, the values of accuracy lies in the range of 56.11% to 79.21% for PitsA, 28.98% to 66.77% for PitsB, 73.73% to 92.09% for PitsC, 86.26% and 96.70% for PitsD, 28.85% to 73.09% for PitsE and 42.34% to 75.10% for PitsF respectively. The maximum and minimum values of F-Measure for the studied projects across all machine learning techniques for different levels of severity are shown in table III. From the experimental study, we have found that accuracy and the difference between maximum and minimum values of F-Measure for the PitsD is very small for severity level 2 and 3. The selected terms will able to distinguish the bug reports into specific class if we have limited class labels. The increase in number of class levels, these terms may not be able to distinguish the bug report into specific class because few of these terms may be common in these class levels. Due to this reason, the best accuracy is obtained for those projects for which only limited class level reports are available. The performance measure accuracy and F-Measure has not been increased significantly for project PitsD as we have increased the number of terms from 25 to 200 for the mentioned machine learning techniques. It means, in case of lesser number of reports, it degrades the performance of machine learning techniques as we increase the number of terms. This is experimentally validated for project PitsD.

Figure 1(a) to figure 1(f) show that the most of the classifiers are stabilizes with 125 or more terms/features

except NB classifier. Accuracy of NB classifier does not get stabilize in most of the projects. The trends with F-Measure of various machine learning techniques of PitsA project are shown in the figure 2. From the figure 2(a) to figure 2(i), these trends depict that the choice of the classifier depends on the number of terms considered to build the classifier.

TABLE II. ACCURACY OF DIFFERENT MACHINE LEARNING TECHNIQUES WITH VARYING NUMBER OF TERMS

Projects	No. of Terms	Accuracy of Machine Learning Techniques								
		NB	*1-NN*	*2-NN*	*3-NN*	*4-NN*	*NBM*	*SVM*	*J4.8*	*RIPPER*
PitsA	25	**56.11%**	73.93%	74.92%	75.58%	76.24%	77.23%	66.34%	62.71%	67.33%
	50	60.73%	74.59%	77.23%	75.58%	**79.21%**	76.57%	65.68%	66.34%	66.01%
	75	65.02%	75.91%	77.56%	75.58%	77.23%	76.57%	65.35%	64.69%	67.66%
	100	68.98%	73.93%	77.23%	75.25%	75.91%	77.23%	66.34%	68.32%	67.00%
	125	73.60%	76.90%	77.23%	78.88%	77.89%	76.90%	66.01%	64.03%	67.66%
	150	76.24%	76.24%	77.23%	77.56%	77.89%	77.23%	67.33%	63.37%	64.69%
	175	77.89%	75.91%	78.22%	77.89%	78.55%	76.57%	68.98%	67.99%	66.67%
	200	78.88%	76.57%	77.89%	79.21%	78.55%	77.56%	69.31%	68.65%	64.69%
PitsB	25	**28.98%**	60.49%	58.76%	60.18%	60.49%	58.76%	57.55%	55.93%	58.66%
	50	38.60%	64.74%	62.92%	64.64%	64.54%	62.11%	60.28%	57.24%	59.78%
	75	49.44%	**66.77%**	64.54%	63.32%	64.54%	63.22%	60.49%	57.14%	59.27%
	100	51.17%	65.25%	63.22%	66.46%	66.46%	64.03%	60.79%	58.36%	59.78%
	125	55.22%	66.26%	62.92%	65.35%	63.83%	64.13%	61.90%	58.76%	60.18%
	150	58.05%	66.36%	63.73%	64.84%	65.35%	64.44%	62.72%	58.87%	60.08%
	175	59.68%	66.16%	63.53%	64.74%	65.45%	64.94%	62.72%	59.37%	59.68%
	200	60.59%	66.06%	64.44%	65.86%	65.45%	65.05%	62.82%	60.28%	59.88%
PitsC	25	77.22%	91.77%	90.19%	**92.09%**	91.46%	85.76%	81.96%	81.33%	81.65%
	50	86.71%	90.19%	88.29%	88.29%	87.97%	85.76%	81.33%	83.86%	77.85%
	75	87.34%	91.14%	88.29%	87.34%	87.34%	83.54%	80.06%	83.23%	77.85%
	100	87.03%	91.14%	87.66%	87.66%	86.71%	84.18%	82.28%	83.23%	79.11%
	125	88.29%	91.77%	88.29%	88.29%	86.39%	84.81%	82.59%	83.54%	78.80%
	150	89.56%	91.14%	87.66%	87.66%	85.76%	86.08%	84.81%	83.86%	**73.73%**
	175	87.97%	90.51%	87.03%	86.71%	85.76%	88.29%	83.86%	81.65%	79.75%
	200	86.71%	89.24%	85.44%	85.44%	84.49%	89.24%	86.08%	80.38%	76.58%
PitsD	25	93.96%	95.05%	95.60%	95.05%	95.05%	91.76%	**96.70%**	96.15%	94.51%
	50	92.31%	93.41%	95.60%	95.05%	95.60%	91.76%	96.15%	95.60%	95.05%
	75	94.51%	93.96%	95.60%	95.60%	95.60%	91.76%	96.15%	95.60%	93.96%
	100	92.86%	93.41%	95.60%	95.60%	95.60%	91.76%	96.15%	95.60%	93.96%
	125	89.56%	93.96%	95.05%	95.05%	95.60%	92.31%	96.15%	95.60%	95.05%
	150	88.46%	94.51%	95.05%	95.05%	95.60%	92.86%	96.15%	95.60%	93.41%
	175	**86.26%**	94.51%	95.05%	95.05%	95.60%	93.41%	96.15%	95.60%	93.96%
	200	87.36%	94.51%	95.05%	95.05%	95.60%	93.41%	96.15%	95.60%	92.86%
PitsE	25	**28.85%**	65.33%	65.58%	66.79%	67.03%	65.33%	69.45%	70.30%	68.97%
	50	33.70%	64.97%	65.82%	65.58%	65.58%	66.79%	69.58%	70.42%	68.97%
	75	38.30%	66.30%	67.39%	65.70%	68.00%	67.39%	69.09%	70.42%	69.21%
	100	43.88%	68.97%	67.52%	68.48%	70.30%	67.76%	69.45%	69.94%	69.21%
	125	43.64%	67.88%	67.88%	70.18%	69.21%	68.61%	69.82%	69.94%	68.97%
	150	48.24%	69.09%	70.30%	70.67%	70.67%	68.73%	69.33%	70.06%	69.21%
	175	49.70%	69.70%	70.06%	71.52%	71.88%	69.45%	69.94%	70.06%	70.06%
	200	51.64%	70.42%	71.64%	73.09%	**73.09%**	69.09%	70.79%	69.82%	69.94%
PitsF	25	**42.34%**	68.51%	71.20%	68.78%	70.79%	65.33%	69.89%	67.34%	66.80%
	50	47.45%	69.31%	74.16%	72.68%	73.35%	66.79%	71.24%	67.34%	67.07%
	75	49.19%	67.83%	72.54%	71.20%	73.22%	67.39%	72.58%	67.34%	66.80%
	100	51.75%	68.64%	72.81%	70.93%	72.41%	67.76%	72.45%	67.34%	66.94%
	125	52.42%	69.31%	73.22%	73.62%	74.02%	68.61%	72.31%	68.55%	68.41%
	150	55.24%	69.58%	72.41%	72.54%	71.87%	68.73%	72.45%	69.49%	69.89%
	175	56.72%	69.85%	73.89%	73.62%	74.02%	69.45%	73.92%	70.03%	70.16%
	200	59.01%	71.06%	**75.10%**	74.29%	73.89%	69.09%	72.98%	69.89%	70.83%

The mentioned machine learning techniques specify the best performance using F-Measure and accuracy when using top 125 terms. These terms are selected using the entropy based information gain measure. The Maximum and minimum values of F-Measure for PitsA project are shown in figure 3. It is clear that severity level 2, and 3 can be predicted with the reasonable value of F-Measure as they show a small amount of difference in maximum and minimum value of F-Measure. Similar trends are also observed in other projects as well.

Figure 3 shows the maximum and minimum value of F-Measure for various severity levels for the PitsA project. Best F-Measure values are obtained for the severity level 2 in all the techniques. Second best has also obtained for the severity level 3 and there is a minor difference between the maximum and minimum values of F-Measure in these techniques except NB classifier. The severity level 4 does not show good performance as there is less number of reports with respect to the number of terms for severity level 4.

From figure 4 of project PitsB, the best values of F-measure are obtained for severity level 3 with all mentioned techniques except NB classifier. In NB classifier, the ranges are quite high for the severity level 3. The second best F-measure has been obtained for severity level 4 except in SVM, J48 and RIPPER.

The best F-Measure for the project PitsC have shown in figure 5 which clearly indicates the maximum values of F-Measure are more than 80% in all techniques for the severity level 3 and severity level 4. PitsD project in figure 6 shows best F-Measure for severity level 3 which is more than 90%. The F-Measure for other severity level i.e., severity level 4 has also been shown as more than 80% except NB and SVM classifier. This may be because there is sufficient number of reports available for these two severity levels i.e., 3 and 4.

For project PitsE in figure 7, the best F-Measure more than 80% for severity level 3 has been obtained except NB classifier. Other severity levels receive mixed performances.

The best F-Measure for severity level 3 has also been observed in project PitsF as shown in figure 8. Other levels of severity receive the varying values of F-Measure.

IV. CONCLUSIONS

Bug severity is the degree of impact that a defect has on the development or operation of a component or system. Bug severity is classified into different levels based on its impact on the system. Summary attribute of the bug reports contains useful and precise information of the bug. In this paper, we have applied various machine learning techniques in determining the severity level of newly reported bugs based on the textual summary of the bug report. We have also computed the value of performance measures namely, precision, recall, F-Measure and accuracy using 5-fold cross validation to validate the applicability of these techniques. It is observed that the performance of machine learning techniques stabilizes as we increase the number of terms from 125 onwards. It is clear that these 125 terms selected using information gain criteria from the term set of the available bug reports will be quite appropriate for the model building. Best F-Measures are received for severity level 2, 3 and 4 by using almost every machine learning technique. Based on the models developed, the severity level for the newly submitted bug reports can be predicted which will be helpful in automatic determination of severity level from 1 to 5. The automatic determination of severity level of the reported bug will be quite useful for the triager in automatic assignment of reported bug to the developer for its fix.

V. FUTURE RESEARCH AGENDA

In future, the authors will work on the following research agenda:

- This study will be carried out on more projects/components of software projects to validate the applicability of machine learning techniques.
- Research will also be carried out using clustering of terms by making a global and local dictionary which can help in reducing the number of features.
- Local dictionaries can be prepared for each severity level by combining domain specific projects.

The analysis can be further carried out to determine the optimum number of bug reports required to get the best performance.

APPENDIX

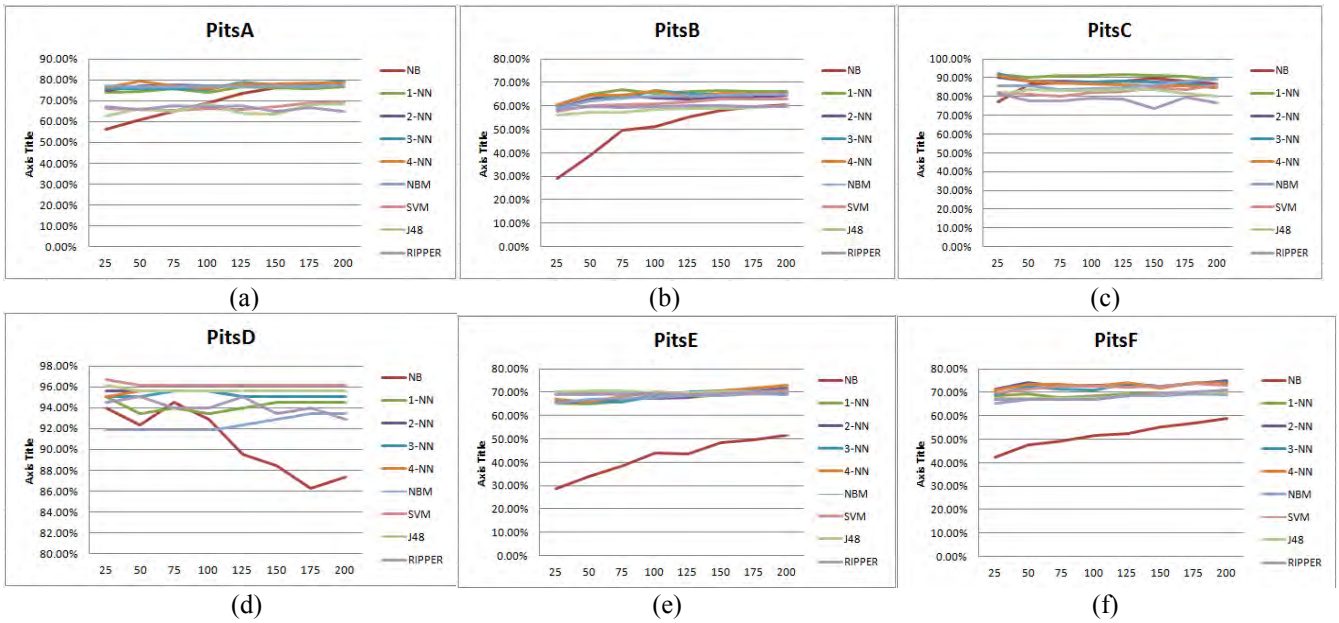

Figure 1. Accuracy of various classifiers for NASA's PITS projects bug repositories.

TABLE III. MAXIMUM AND MINIMUM VALUES OF F-MEASURE FOR PITS PROJECTS WITH VARIOUS MACHINE LEARNING TECHNIQUES

Projects →		**PitsA**		**PitsB**		**PitsC**		**PitsD**		**PitsE**		**PitsF**	
Algorithm	***Severity***	***Maximum***	***Minimum***	***Maximum***	***Minimum***	***Maximum***	***Minimum***	***Maximum***	***Minimum***	***Maximum***	***Minimum***	***Maximum***	***Minimum***
NB	pred. 3	80.98%	51.95%	74.19%	44.73%	89.58%	85.26%	96.99%	92.26%	65.03%	44.10%	66.83%	53.54%
	pred. 2	87.62%	83.87%	37.04%	21.06%	-	-	-	-	12.58%	7.49%	21.05%	8.89%
	pred. 5	-	-	33.33%	14.74%	42.86%	16.66%	-	-	26.15%	19.81%	40.46%	27.66%
	pred. 4	43.75%	28.23%	52.28%	21.17%	91.41%	82.25%	73.34%	41.03%	48.36%	30.42%	56.12%	41.96%
1-NN	pred. 3	78.57%	74.83%	75.87%	72.61%	91.41%	82.25%	97.33%	96.41%	79.77%	76.26%	79.56%	77.84%
	pred. 2	87.89%	83.40%	53.85%	47.27%	-	-	-	-	20.83%	9.52%	66.67%	53.33%
	pred. 5	-	-	34.71%	14.89%	71.43%	62.50%	-	-	57.50%	32.84%	44.90%	14.81%
	pred. 4	46.38%	28.07%	63.17%	42.03%	93.37%	90.75%	74.07%	66.67%	57.68%	36.16%	59.33%	43.21%
2-NN	pred. 3	82.41%	77.76%	76.64%	71.94%	89.93%	85.71%	97.63%	97.33%	82.34%	77.93%	83.75%	81.67%
	pred. 2	86.79%	79.93%	45.46%	32.00%	-	-	-	-	19.61%	14.29%	62.50%	50.00%
	pred. 5	-	-	31.34%	12.50%	46.16%	40.00%	-	-	50.00%	28.95%	48.42%	16.44%
	pred. 4	28.00%	5.53%	51.76%	34.84%	92.08%	87.04%	75.00%	72.00%	49.73%	30.32%	56.97%	44.07%
3-NN	pred. 3	81.48%	78.05%	74.52%	71.43%	91.05%	85.30%	97.63%	97.31%	82.16%	76.98%	81.87%	79.56%
	pred. 2	89.69%	85.72%	32.26%	23.26%	-	-	-	-	24.39%	6.46%	62.50%	57.14%
	pred. 5	-	-	33.33%	6.81%	76.92%	20.00%	-	-	47.76%	28.98%	50.00%	20.69%
	pred. 4	33.90%	12.24%	60.63%	45.75%	93.37%	87.79%	75.00%	74.07%	58.83%	40.11%	60.76%	43.51%
4-NN	pred. 3	83.08%	79.66%	76.50%	72.08%	90.49%	85.22%	97.63%	97.31%	82.41%	78.09%	82.95%	81.12%
	pred. 2	88.99%	84.51%	35.30%	24.69%			-	-	27.78%	6.90%	62.50%	53.33%
	pred. 5			32.00%	11.36%	76.92%	20.00%	-	-	43.24%	30.00%	48.89%	19.67%
	pred. 4	28.00%	4.65%	57.63%	44.29%	92.70%	85.97%	76.92%	76.92%	57.69%	37.43%	58.38%	46.88%
NBM	pred. 3	81.50%	80.55%	75.63%	72.49%	89.06%	80.34%	96.53%	95.70%	80.84%	78.72%	80.84%	78.72%
	Pred. 2	84.88%	83.00%	-	-	-	-	-	-	-	-	-	-
	pred. 5	-	-	-	-	-	-	-	-	4.76%	4.76%	4.76%	4.76%
	pred. 4	-	-	54.12%	30.77%	91.06%	86.96%	88.66%	14.28%	41.50%	16.80%	41.50%	16.80%
SVM	pred. 3	76.81%	74.57%	74.23%	71.69%	82.46%	73.93%	96.99%	92.26%	81.56%	80.28%	83.30%	80.89%
	pred. 2	65.46%	55.85%	68.42%	33.33%	-	-	-	-	15.38%	13.79%	61.53%	57.14%
	pred. 5	-	-	12.70%	6.35%	85.71%	40.00%	-	-	54.23%	32.65%	25.45%	15.09%
	pred. 4	20.51%	5.40%	38.98%	20.53%	89.11%	59.76%	73.34%	41.03%	35.41%	30.98%	47.52%	36.51%
J48	pred. 3	74.57%	69.27%	81.84%	70.11%	83.39%	77.52%	97.89%	97.60%	81.36%	81.01%	81.00%	79.67%
	pred. 2	72.94%	64.92%	48.48%	39.50%	-	-	-	-	17.65%	6.67%	61.53%	61.53%
	pred. 5	-	-	3.17%	2.19%	46.16%	25.01%	-	-	36.36%	21.81%	0.00%	0.00%
	pred. 4	19.61%	4.35%	37.92%	16.28%	87.03%	82.61%	82.76%	78.57%	44.70%	38.85%	40.96%	18.19%
RIPPER	pred. 3	74.67%	72.04%	72.65%	71.48%	82.67%	71.64%	97.34%	96.12%	81.36%	80.43%	81.28%	79.36%
	pred. 2	-	-	56.41%	40.00%	-	-	-	-	12.90%	6.90%	53.33%	42.85%
	pred. 5	-	-	11.94%	6.06%	40.00%	18.19%	-	-	33.90%	25.00%	28.58%	28.58%
	pred. 4	9.75%	4.76%	35.79%	28.27%	82.50%	77.05%	69.57%	58.34%	46.02%	35.18%	36.43%	16.59%

- Does not show the performance of this severity level

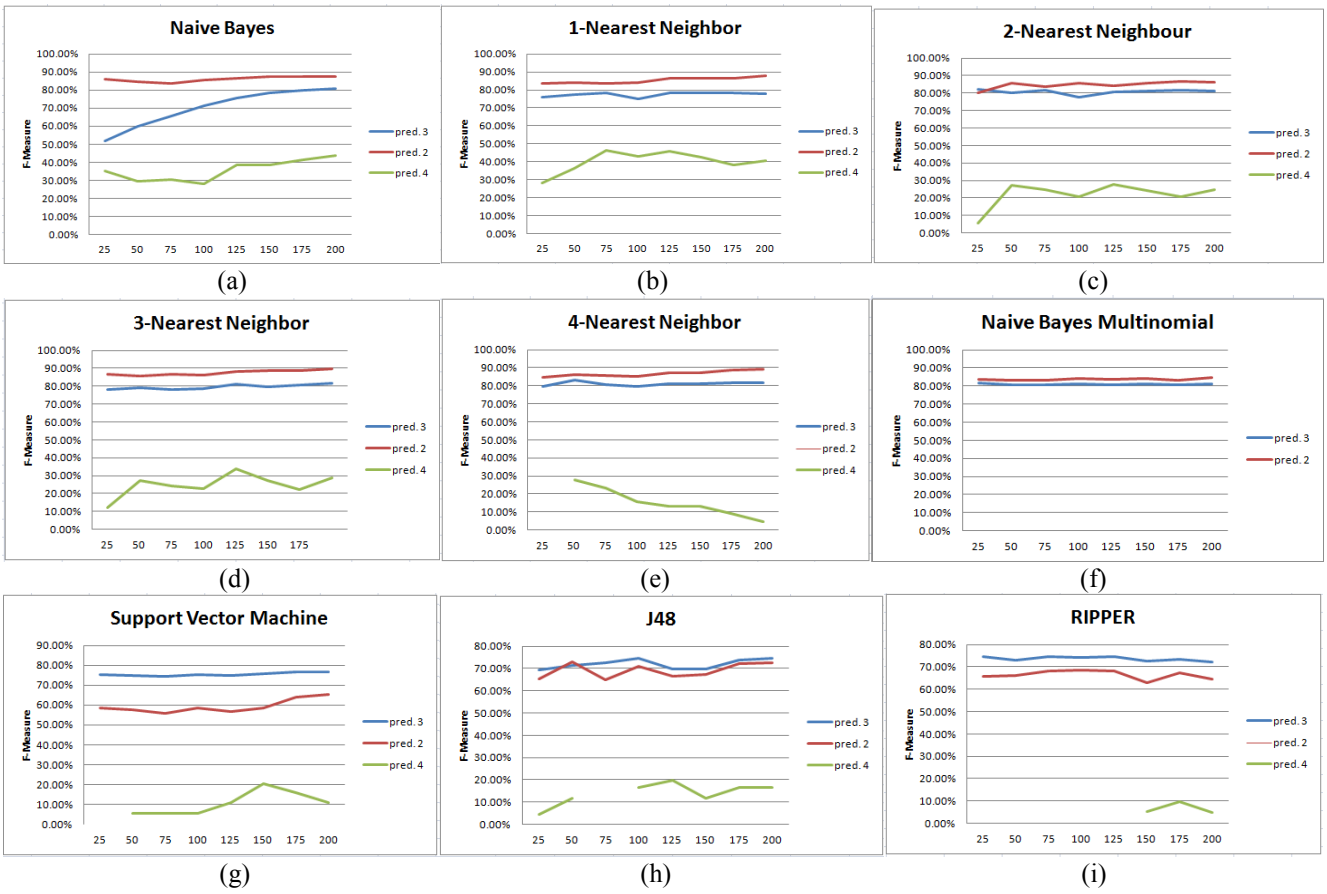

Figure 2. The F-Measure for different severity levels of NASA's PitsA project with varying number of terms.

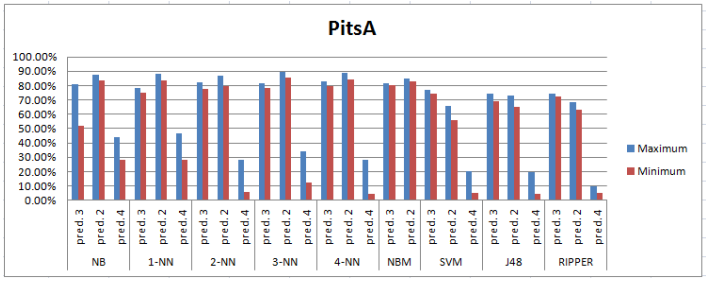

Figure 3. The F-Measure for different predicted (pred.) severity levels of NASA's PitsA project with varying number of terms.

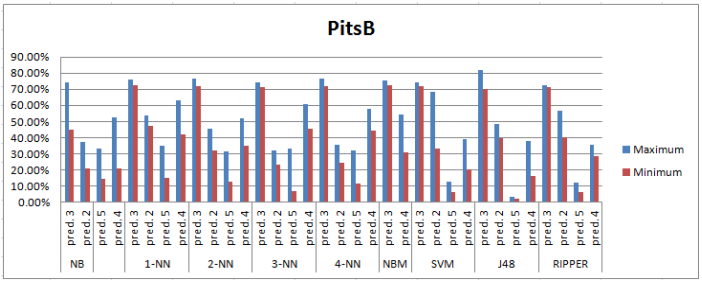

Figure 4. The F-Measure for different predicted (pred.) severity levels of NASA's PitsB project with varying number of terms.

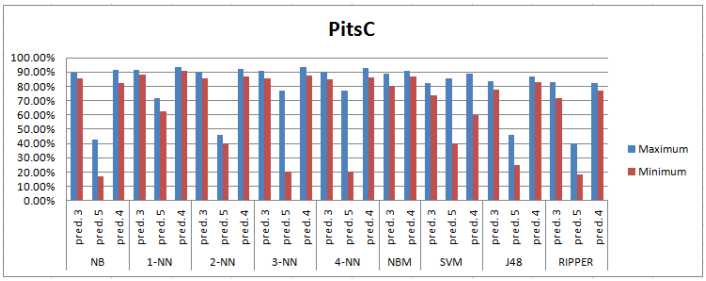

Figure 5. The F-Measure for different predicted (pred.) severity levels of NASA's PitsC project with varying number of terms.

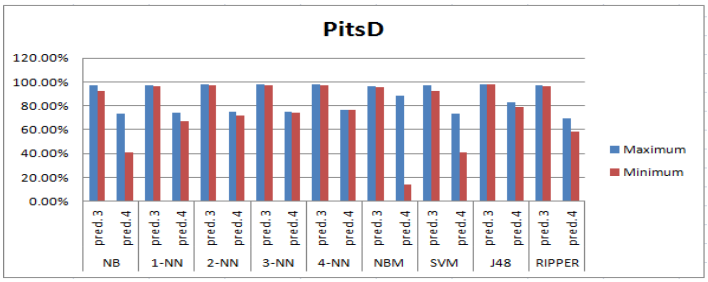

Figure 6. The F-Measure for different predicted (pred.) severity levels of NASA's PitsD project with varying number of terms.

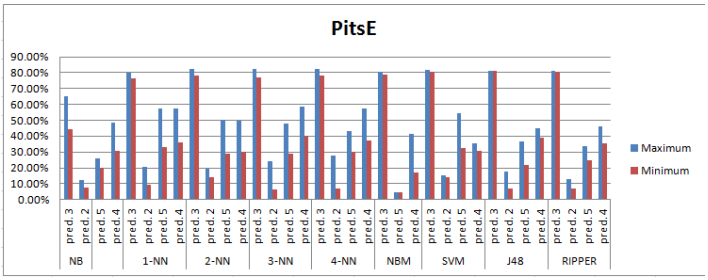

Figure 7. The F-Measure for different predicted (pred.) severity levels of NASA's PitsE project with varying number of terms.

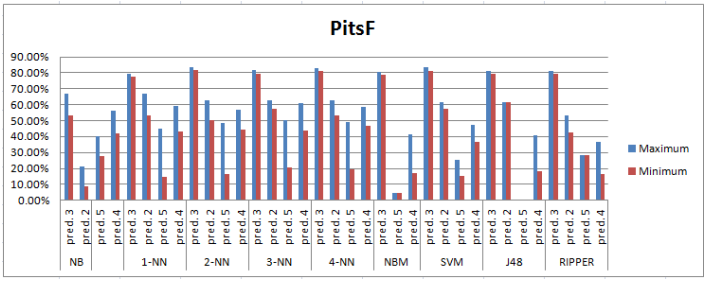

Figure 8. The F-Measure for different predicted (pred.) severity levels of NASA's PitsF project with varying number of terms.

ACKNOWLEDGMENT

The Authors are thankful to Prof. P.K. Kapur and Dr. Punam Bedi for their technical support. The authors are also thankful to reviewers for their valuable comments in revising the manuscript.

REFERENCES

[1] G. Boetticher, T. Menzies and T. Ostrand. PROMISE Repository of empirical software engineering data. http://promisedata.org/ repository, West Virginia University, Department of Computer Science, 2007.

[2] G. Antoniol, M. K. Ayari, F.K. Di Penta and Y. G. Gueh´eneuc. Is it a bug or an enhancement?: a text-based approach to classify change requests. In *CASCON '08: Proceedings of the conference of the center for advanced studies on collaborative research*. ACM, 2008, pp. 304–318.

[3] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *IEEE International Conference on Software Maintenance*, Oct. 2008, pp. 346–355.

[4] A. Lamkanfi, S. Demeyer, E. Giger and B. Goethals. Predicting the severity of a reported bug. In Mining Software Repositories (MSR) 2010: 1-10.

[5] A. Lamkanfi, S. Demeyer, Q.D. Soetens and T. Verdonck. Comparing mining algorithms for predicting the severity of a reported bug. CSMR 2011: 249-258

[6] I. Herraiz, D. M. German, J.M. Gonzalez-Barahona, and G. Robles. Towards a simplification of the bug report form in Eclipse. In *5th International Working Conference on Mining Software Repositories*, May 2008.

[7] M. Porter. An algorithm for suffix stripping. *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[8] R. Feldman and J. Sanger. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, December 2006.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[10] N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. ACM, 2008, pp. 308–318.

[11] X. Wang, L. Zhang, T. Xie, J. Anvik and J. Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proceedings 30th International Conference on Software Engineering (ICSE'08)*, Leipzig, Germany, 10 - 18 May 2008.

[12] P. Runeson, M. Alexandersson and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th international conference on Software Engineering*, 2007.

[13] D. Cubranic and G.C. Murphy. Automatic bug triage using text categorization. In *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*, June 2004, pp. 92–97.

[14] J. Anvik, L. Hiew, and G.C. Murphy. Who should fix the bug? In Proceedings of the 28th International conference on Software engineering, 2006.

[15] J. Gaeul, K. Sunghun, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *Proceedings of the European Software Engineering Conference 2009*. ACM, 2009, pp. 111–120.

[16] V.B. Singh and K.K. Chaturvedi. Bug tracking and reliability assessment system. In *International Journal of Software Engineering and Its Applications*. 2011, Vol 5(4). PP. 17-30.

[17] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. University at Dortmund, LS VIII-Report, Tech. Rep. 23, 1997.

[18] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz and T. Euler. *YALE: Rapid Prototyping for Complex Data Mining Tasks*, in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006. (http://www.rapid-i.com)

[19] P.J. Guo, T. Zimmermann, N. Nagappan and B. Murphy. Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows. In ICSE, 2010.

[20] P. Bhattacharya and I. Neamtiu. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In ICSM, 2010.

[21] D. Matter, A. Kuhn, and O. Nierstrasz. Assigning bug reports using a vocabulary based expertise model of developers. In MSR 2009.

[22] T. M. Mitchell. Machine Learning. McGraw Hill. 1997.

[23] C. W. Hsu and C. J. Lin. A comparison of methods for multi-class support vector machines. IEEE Transactions on Neural Networks, 13(2002), 415-425.

[24] W. W. Cohen. Fast effective rule induction. In Proceedings 12th International Conference on Machine Learning. pp. 115-123. 1995.

[25] J. C. Platt, N. Cristianini and J. Shawe-Taylor. Large margin DAGs for multiclass classification. Advances of neural information processing systems, Vol. 12. pp. 547-553. MIT Press. 2000.

[26] M. Konchady. Text Mining Application Programming. Thomson Delmar Learning. 2006

[27] S.M. Weiss, N. Indurkhya, T. Zhang and F.J. Damerau. Text Mining: Predictive Methods for Analyzing Unstructured Information. Springer 2005.

[28] V. Vapnik. The Nature of Statistical Learning Theory. Berlin, Springer-Verlag. 1995.

[29] A. Sureka and P. Jalote. Detecting Duplicate Bug Report Using Character N-Gram-Based Features. APSEC, 2010, pp. 366-374.

[30] P.K. Kapur, A.G. Aggarwal and A. Tandon. Two Dimensional Software Reliability Growth Model with Faults of Different Severity. Communications in Dependability and Quality Management. 2010. Vol 13(3). pp. 98-110.

[31] J. R. Quinlan. Induction of Decision Trees. Machine Learning. Vol. 1(1). pp. 81-106. 1986.

[32] J. R. Quinlan. C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA. 1993.

BIOGRAPHIES

K. K. Chaturvedi is a Scientist, Computer Applications at Indian Agricultural Statistics Research Institute (Indian Council of Agricultural Research), New Delhi, India. He published around fifteen papers in various journals and conferences. His research interests include application of Data Warehousing, Data Mining, Software Configuration Management and Mining Software Repositories. He is currently doing Ph.D. in Computer Science from Department of Computer Science, University of Delhi, Delhi, India.

V. B. Singh is currently working as Associate Professor in the Department of Computer Science at Delhi College of Arts and Commerce, University of Delhi, Delhi (India). He received his M.C.A. degree from M. M. M. Engineering College, Gorakhpur, U.P. (India) and Ph.D. degree from University of Delhi in Software Reliability. He has published more than twenty five research papers. His research interests include mining software repositories and software reliability engineering. He is member of ACM.