Multiattribute Based Machine Learning Models for Severity Prediction in Cross Project Context

Meera Sharma[1], Madhu Kumari[2], R.K. Singh[3], and V.B. Singh[4]

[1] Department of Computer Science,
University of Delhi, Delhi, India
[2] Delhi College of Arts & Commerce,
University of Delhi, Delhi, India
[3] Department of Information Technology,
Indira Gandhi Delhi Technical University for Women, Delhi, India
[4] Delhi College of Arts & Commerce, University of Delhi, Delhi, India
{meerakaushik,mesra.madhu,rksingh988,vbsinghdcacdu}@gmail.com

Abstract. The severity level of a reported bug is an important attribute. It describes the impact of a bug on functionality of the software. In the available literature, machine learning techniques based prediction models have been proposed to assess the severity level of a bug. These prediction models have been developed by using summary of a reported bug i.e. the description of a bug reported by a user. This work has been also extended in cross project context to help the projects whose historical data is not available. Till now, the literature reveals that bug triager assess the severity level based on only the summary report of a bug but we feel that the severity level of a bug may change its value during the course of fixing and moreover, the severity level is not only characterized by the summary of bug report but also by other attributes namely priority, number of comments, number of dependents, number of duplicates, complexity, summary weight and cc list. In this paper, we have developed prediction models for determining the severity level of a reported bug based on these attributes in cross project context. For empirical validation, we considered 15,859 bug reports of Firefox, Thunderbird, Seamonkey, Boot2Gecko, Add-on SDK, Bugzilla, Webtools and addons.mozilla.org products of Mozilla open source project to develop the classification models based on Support Vector Machine (SVM), Naïve Bayes (NB) and K-Nearest Neighbors (KNN).

1 Introduction

The dependence on software is almost inescapable in every sphere of human activities. The developing countries are also getting an edge in the usages of software and especially open source software. During the last four decades, various researchers have developed methods and tools to improve the quality of software. Recently, due to availability of various software repositories namely source code, bugs, attributes of bugs, source code changes, developer communication and mailing list, new research areas in software engineering have been emerged like mining software repositories, empirical software engineering and machine learning based software engineering. Various machine learning based prediction models have been developed and is currently being used to improve the quality of software in terms of choosing right developer to fix the bugs, predicting bug fix time, predicting the attributes of a bug namely severity and priority, and bugs lying dormant in the software [1-6]. A large number of bug tracking systems have been developed which help in recording and solving the bugs faced by users. A bug is characterized by many attributes, and severity is one of them. The degree of impact of a bug on the functionality of the software is known as its severity. It is typically measured according to different levels from 1(blocker) to 7(trivial). The summary attribute of a bug report consists of the brief description about the bug. Studies have been carried to predict the bugs either in sever or non-sever category and in different levels using different machine learning techniques [1, 2, and 3]. Recently, an effort has been made to conduct an empirical comparison of different machine learning techniques namely Support Vector Machine, Probability based Naïve Bayes, Decision tree based J48, RIPPER and Random Forest in predicting the bug severity of open and closed source projects [4 and 5]. In all these works, the classifiers have been developed based on the past data of the same project. But, in many cases where the project is new and for which the historical data is not available, it is difficult to predict the severity level of bugs reported in those projects. In this scenario, cross project study can help. An attempt has been made to predict the severity levels of bugs in cross project context on a limited data set using SVM [11].

A software bug report is characterized by the following attributes shown in table 1[12].

In this paper, we have made an attempt to predict the severity level of a reported bug by using multiple attributes namely priority, bug fix time, number of comments, number of bugs on which it is dependent, number of duplicates for it, number of members in cc list, summary weight and complexity of bug in cross project context. Summary weight and complexity are two new attributes which we have derived and discussed in section 2 of the paper. To study and investigate the severity prediction and identification of training candidates in cross project context, we have proposed the following research question:

Research Question 1: What is the predictive level that can be achieved by multi attribute based cross project severity prediction models?

In answer of this question, we found that across all the classifiers, performance measures namely accuracy, precision, recall and f-measure lie in the range of 37.34 to 91.63%, 94.99 to 100%, 44.88 to 97.86% and 61.18 to 95.99% respectively for multiattribute based cross project severity prediction.

The rest of the paper is organized as follows. Section 2 of the paper describes the datasets and prediction models. Results have been presented in section 3. Section 4 presents the related work. Threats to validity have been discussed in section 5 and finally the paper is concluded in section 6.

Table 1. Bug Attributes description

Attribute	Short description
Severity	This indicates how severe the problem is. e.g. trivial, critical, etc.
Bug Id	The unique numeric id of a bug.
Priority	This field describes the importance and order in which a bug should be fixed compared to other bugs. P1 is considered the highest and P5 is the lowest.
Resolution	The resolution field indicates what happened to this bug. e.g. FIXED
Status	The Status field indicates the current state of a bug. e.g. NEW, RESOLVED
Number of Comments	Bugs have comments added to them by users. #comments made to a bug report.
Create Date	When the bug was filed.
Dependencies	If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.
Summary	A one-sentence summary of the problem.
Date of Close	When the bug was closed.
Keywords	The administrator can define keywords which you can use to tag and categorize bugs e.g. the Mozilla project has keywords like crash and regression.
Version	The version field defines the version of the software the bug was found in.
CC List	A list of people who get mail when the bug changes. #people in CC list.
Platform and OS	These indicate the computing environment where the bug was found.
Number of Attachments	Number of attachments for a bug.
Bug Fix Time	Last Resolved time-Opened time. Time to fix a bug.

2 Description of Data Sets and Prediction Models

In this paper, an empirical experiment has been conducted on 15,859 bug reports of the Mozilla open source software products namely Firefox, Thunderbird, Seamonkey, Boot2Gecko, Add-on SDK, Bugzilla, Webtools and Addons.mozilla.org. We collected bug reports for resolution “fixed” and status “verified”, “resolved” and “closed” because only these types of bug reports contain the meaningful information

Table 2. Number of bug reports in each product

Product	Number of bugs	Observation period
Firefox	2712	Apr. 2001-Aug. 2013
Thunderbird	115	Apr. 2000-Mar. 2013
Seamonkey	6012	Apr. 1998-July 2013
Boot2Gecko	1932	Feb. 2012-Aug. 2013
Add-on SDK	616	May 2009-Aug. 2013
Webtools	366	Oct. 1998-Aug. 2013
Bugzilla	964	Sept. 1994-June 2013
Addons.mozilla.org	3142	Oct. 2003-Aug. 2013

for building and training the models. The collected bug reports from Bugzilla have also been compared and validated against general change data (i.e. CVS or SVN records). Table 2 shows the data collection in the observed period.

We have used 9 quantified bug attributes namely severity, priority, bug fix time, number of comments, number of bugs on which it is dependent, number of duplicates for it, number of members in cc list, summary weight and complexity of the bug. Severity, Priority, Depends on, Duplicate count and Bug complexity are categorical in nature and rests of the attributes are continuous.

Bug fix time has been calculated as Last_resolved time-Opened time for a bug. A large variation in bug fix time can affect the result. That's why we have considered bugs with fix time less than or equal to 99 days (as maximum number of bugs had fix time in this range). Depending on bug fix time we derived complexity of the bug by defining three levels for fix time. We observed that the bugs are categorized in three levels depending upon their bug fix time and three categories of bug fix time are 0-32 days level 1, 33-65 days level 2 and 66-99 days level 3. Here, we define the complexity of bug from bug fixer point of view. There is a need to quantify summary attribute. This has been done with the help of a two-step process: (1) text mining was applied to get the weight of individual terms of summary by using information gain criteria with the help of a process developed in RapidMiner tool (2) weights of distinct terms in a summary are added to get the weight of the bug summary. There are very less number of reports with number of dependent bugs as most of these bugs are independent. If the bug is independent, the parameter is set to 0 otherwise 1. Similar rule we have followed for duplicate count attribute.

We have carried out following steps for our study:

1. Data Extraction

a. Download the bug reports of different products of Mozilla open source software from the CVS repository: https://bugzilla.mozilla.org/
b. Save bug reports in excel format.

2. Data Pre-processing

a. Set fix time = Last_resolved time-Opened_ time. Filter the bug reports for fix time less than equal to 99 days.
b. Set all non-zero values to 1 for depends on and duplicate count attributes.

3. Data Preparation

a. Derive bug complexity by categorizing bugs in three categories of bug fix time 0-32, 33-65 and 66-99 days.

b. Calculate the summary weight by using information gain criteria in RapidMiner tool.

4. Modeling

a. Build three models based on SVM, NB and KNN classifiers.

b. Train the model by using eight attributes namely priority, bug fix time, number of comments, number of bugs on which it is dependent, number of duplicates for it, number of members in cc list, summary weight and complexity of the bug to predict the bug severity.

c. Test the model for another dataset.

5. Testing and Validation

a. Assess the performance of prediction models.

We conducted our study by using Statistica and RapidMiner software tools.

3 Results and Discussion

We have applied 3 models based on SVM, NB and KNN classifiers in Statistica software to predict severity in cross project context. In all tables '–' shows that no experiment has been done for that particular combination of testing and training dataset because both training and testing datasets are same.

Table 3. Cross project severity prediction accuracy of SVM

Training Dataset	Testing Dataset (Accuracy in %)							
	Seamonkey	Thuderbird	Firefox	Boot2Gecko	Bugzilla	AddOnSDK	Addons.mozilla. org	Webtools
Seamonkey	-	59.13	76.21	53.10	34.12	46.91	61.13	63.38
Thuderbird	62.09	-	82.78	83.69	37.34	91.23	80.42	73.77
Firefox	91.23	64.34	-	83.69	37.34	91.23	80.42	73.77
Boot2Gecko	62.09	64.91	82.77	-	37.38	91.23	80.42	73.97
Bugzilla	58.88	53.04	60.28	45.23	-	48.86	49.26	58.19
AddOnSDK	62.09	64.34	82.78	83.69	37.34	-	80.42	73.77
Addons.mozilla. org	62.09	64.34	82.78	83.69	37.34	91.23	-	73.77
Webtools	62.09	64.34	82.78	83.69	37.34	91.23	80.42	-

Table 3 shows cross project severity prediction accuracy of SVM based model for different training data sets. Across all datasets Firefox is the best training candidate for Seamonkey, Boot2Gecko, AddOnSDK and Addons.mozilla.org datasets. Boot2Gecko is the best training candidate for Thunderbird, Bugzilla and Webtools datasets. We observed that in SVM for different training datasets we are getting same accuracy.

Table 4. Cross project severity prediction accuracy of NB

Training Dataset	Testing Dataset (Accuracy in %)							
	Seamonkey	Thuderbird	Firefox	Boot2Gecko	Bugzilla	AddOnSDK	Addons.mozilla.org	Webtools
Seamonkey	-	45.22	78.69	45.91	29.98	37.5	46.44	51.37
Thuderbird	62.06	-	82.71	83.02	37.24	89.12	78.58	73.5
Firefox	63.07	59.13	-	77.74	35.48	68.51	70.81	69.13
Boot2Gecko	63.15	65.18	83.89	-	45.01	91.63	82.46	78.26
Bugzilla	59.03	47.83	62.72	77.75	-	84.42	77.75	71.86
AddOnSDK	-	-	-	-	-	-	-	-
Addons.mozilla.org	60.58	58.26	73.19	59.42	37.55	88.8	-	72.95
Webtools	61.04	63.48	78.47	74.95	38.07	90.1	79.98	-

Table 4 shows cross project severity prediction accuracy of NB based model. Across all datasets Boot2Gecko is the best training candidate for Seamonkey, Thunderbird, Firefox, Bugzilla, Webtools, AddOnSDK and Addons.mozilla.org datasets. Thunderbird is the best training candidate for Boot2Gecko dataset. For AddOnSDK dataset, training model was giving a zero standard deviation for the conditional distribution of the independent variable CC Count. So, we were not able to run the experiment for this dataset as training candidate.

Table 5. Cross project severity prediction accuracy of KNN

Training Dataset	Testing Dataset (Accuracy in %)							
	Seamonkey	Thuderbird	Firefox	Boot2Gecko	Bugzilla	AddOnSDK	Addons.mozilla.org	Webtools
Seamonkey	-	44.35	66.74	53.31	34.02	72.4	67.95	66.39
Thuderbird	50.7	-	62.5	56.99	33.51	71.59	62.92	57.38
Firefox	61.33	60	-	76.6	37.03	89.77	79.44	73.22
Boot2Gecko	61.91	63.72	81.37	-	43.77	90.97	0.82	77.68
Bugzilla	36.64	36.52	38.27	28.99	-	44.64	44.14	39.62
AddOnSDK	61.82	64.35	81.27	79.76	37.45	-	80.17	72.68
Addons.mozilla.org	59.51	57.39	73.93	68.94	36.51	86.2	-	72.4
Webtools	56.95	52.17	69.28	63.56	36.41	79.06	71.16	-

Table 5 shows cross project severity prediction accuracy of KNN based model.

Across all datasets Boot2Gecko is the best training candidate for Seamonkey, Firefox, Bugzilla, AddOnSDK and Webtools datasets. AddOnSDK is the best training candidate for Thunderbird, Boot2Gecko and Addons.mozilla.org datasets.

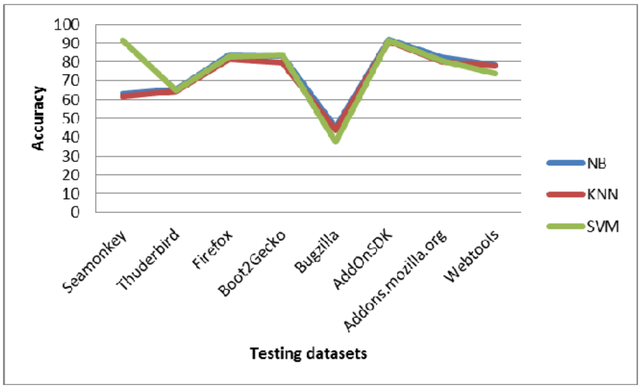

Fig. 1. Cross project accuracy (%) for best training candidates in NB, KNN and SVM

Figure 1 shows the maximum cross project accuracy of all testing datasets for the best training candidates in NB, KNN and SVM.

In order to evaluate different performance measures namely precision, recall and f-measure of different classifiers, we have considered severity level 4 only. Due to less number of reports for other severity levels compared to severity level 4, we are getting low performance for these severity levels. This is one of the problems in multi-class prediction where we have imbalance data sets. For empirical validation, we have considered the performance of different machine learning techniques only for severity level 4.

In case of NB classifier, the testing dataset Seamonkey has precision in the range of 92.69 to 100% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 100%. The testing dataset Thunderbird has precision in the range of 59.46 to 98.65% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 98.65%. The testing dataset Firefox has precision in the range of 73.85 to 99.91% for different training candidates. For this testing dataset Thunderbird as training candidate has given the maximum precision which is 99.91%. The testing dataset Boot2Gecko has precision in the range of 49.04 to 99.2% for different training candidates. For this testing dataset Thunderbird as training candidate has given the maximum precision which is 99.2%. The testing dataset Bugzilla has precision in the range of 60.83 to 100% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 100%. The testing dataset AddOnSDK has precision in the range of 39.32 to 98.93% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 98.93%. The testing dataset Addons.mozilla.org has precision in the range of 55.4 to 99.96% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 99.96%. The testing dataset Webtools has precision in the range of 66.3 to 100% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 100%.

In case of KNN classifier the testing dataset Seamonkey has precision in the range of 50.66 to 99.28% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 99.28%. The testing dataset Thunderbird has precision in the range of 48.65 to 98.65% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 98.65%. The testing dataset Firefox has precision in the range of 43.43 to 98.04% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 98.04%. The testing dataset Boot2Gecko has precision in the range of 31.54 to 94.99% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 94.99%. The testing dataset Bugzilla has precision in the range of 78.33 to 99.17% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 99.17%. The testing dataset AddOnSDK has precision in the range of 47.69 to 98.4% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum precision which is 98.4%. The testing dataset Addons.mozilla.org has precision in the range of 52.28 to 99.68% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum precision which is 99.68%. The testing dataset Webtools has precision in the range of 50 to 99.26% for different training candidates. For this testing dataset Firefox as training candidate has given the maximum precision which is 99.26%.

In case of SVM classifier the testing dataset Seamonkey has precision in the range of 87.38 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla as training candidates have given the maximum precision which is 100%. The testing dataset Thunderbird has precision in the range of 70.27 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%. The testing dataset Firefox has precision in the range of 70.47 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%. The testing dataset Boot2Gecko has precision in the range of 50.34 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%. The testing dataset Bugzilla has precision in the range of 77.5 to 100% for different training candidates. For this testing dataset all datasets except Seamonkey as training candidates has given the maximum precision which is 100%. The testing dataset AddOnSDK has precision in the range of 48.75 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%. The testing dataset Addons.mozilla.org has precision in the range of 57.42 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%.

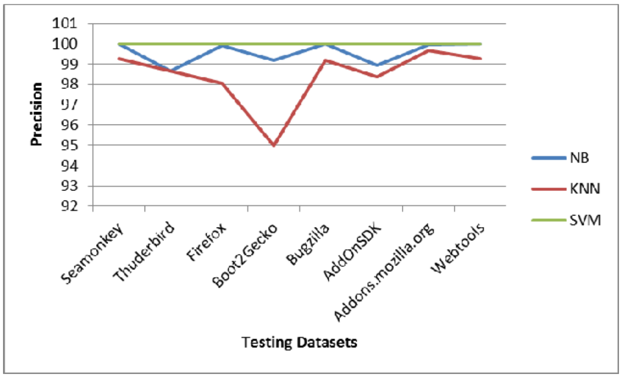

Fig. 2. Cross project precision (%) for best training candidates in NB, KNN and SVM

The testing dataset Webtools has precision in the range of 74.07 to 100% for different training candidates. For this testing dataset all datasets except Bugzilla and Seamonkey as training candidates have given the maximum precision which is 100%.

Figure 2 shows the maximum cross project precision of all testing datasets for the best training candidates in NB, KNN and SVM.

In case of NB classifier, the testing dataset Seamonkey has recall in the range of 62.25 to 64.53% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 64.53%. The testing dataset Thunderbird has recall in the range of 65.31 to 86.27% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 86.27%. The testing dataset Firefox has recall in the range of 82.89 to 85.95% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 85.95%. The testing dataset Boot2Gecko has recall in the range of 81.28 to 91.25% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 91.25%. The testing dataset Bugzilla has recall in the range of 37.86 to 45% to for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum recall which is 45%. The testing dataset AddOnSDK has recall in the range of 91.28 to 97.79% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 97.79%. The testing dataset Addons.mozilla.org has recall in the range of 80.54 to 86.69% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 86.69%. The testing dataset Webtools has recall in the range of 73.76 to 82.87% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 82.87%.

In case of KNN classifier the testing dataset Seamonkey has recall in the range of 62.17 to 64.87% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 64.87%. The testing dataset Thunderbird has recall in the range of 63.64 to 73.47% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 73.47%. The testing dataset Firefox has recall in the range of 82.99 to 86.51% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 86.51%. The testing dataset Boot2Gecko has recall in the range of 83.46 to 88.22% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 88.22%. The testing dataset Bugzilla has recall in the range of 37.42 to 44.88% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum recall which is 44.88%. The testing dataset AddOnSDK has recall in the range of 89.93 to 93.62% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 93.62%. The testing dataset Addons.mozilla.org has recall in the range of 80.31 to 82.61% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum recall which is 82.61%. The testing dataset Webtools has recall in the range of 73.14 to 78.3% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum recall which is 78.3%.

In case of SVM classifier the testing dataset Seamonkey has recall in the range of 62.09 to 91.23% for different training candidates. For this testing dataset Firefox as training candidate has given the maximum recall which is 91.23%. The testing dataset Thunderbird has recall in the range of 64.35 to 84% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 84%. The testing dataset Firefox has recall in the range of 82.78 to 86.97% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 86.97%. The testing dataset Boot2Gecko has recall in the range of 82.42 to 91.16% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 91.16%. The testing dataset Bugzilla has recall in the range of 37.34 to 44.89% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum recall which is 44.89%. The testing dataset AddOnSDK has recall in the range of 91.23 to 97.86% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum recall which is 97.86%. The testing dataset Addons.mozilla.org has recall in the range of 80.43 to 85.1% for different training candidates. For this testing dataset Bugzilla as training candidate has given the maximum recall which is 85.1%. The testing dataset Webtools has recall in the range of 73.77 to 91.23% for different training candidates. For this testing dataset Thunderbird as training candidate has given the maximum recall which is 91.23%.

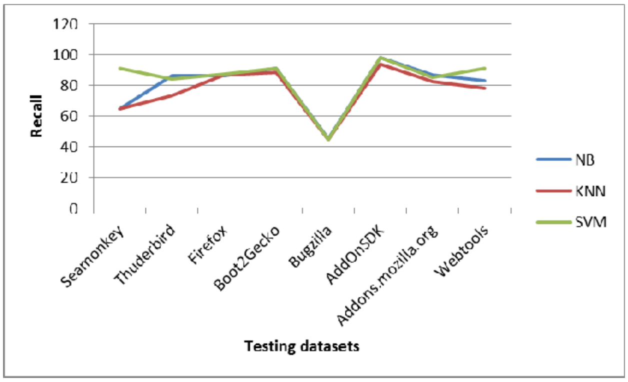

Fig. 3. Cross project recall (%) for best training candidates in NB, KNN and SVM

Figure 3 shows the maximum cross project recall of all testing datasets for the best training candidates in NB, KNN and SVM.

In case of NB classifier, the testing dataset Seamonkey has f-measure in the range of 76.09 to 77.8% for different training candidates. For this testing dataset Firefox as training candidate has given the maximum f-measure which is 77.8%. The testing dataset Thunderbird has f-measure in the range of 69.28 to 78.92% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 78.92%. The testing dataset Firefox has f-measure in the range of 79.44 to 91.33% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 91.33%. The testing dataset Boot2Gecko has f-measure in the range of 63.8 to 90.8% for different training candidates. For this testing dataset Thunderbird as training candidate has given the maximum f-measure which is 90.8%. The testing dataset Bugzilla has f-measure in the range of 50.52 to 62.07% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 62.07%. The testing dataset AddOnSDK has f-measure in the range of 56.09 to 95.94% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 95.94%. The testing dataset Addons.mozilla.org has f-measure in the range of 67.6 to 90.44% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 90.44%. The testing dataset Webtools has f-measure in the range of 73.66 to 87.8% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 87.8%.

In case of KNN classifier the testing dataset Seamonkey has f-measure in the range of 56.89 to 77.04% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 77.04%. The testing dataset Thunderbird has f-measure in the range of 58.54 to 78.26% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 78.26%. The testing dataset Firefox has f-measure in the range of 57.83 to 90.13% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 90.13%. The testing dataset Boot2Gecko has f-measure in the range of 46.32 to 88.86% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum f-measure which is 88.86%. The testing dataset Bugzilla has f-measure in the range of 52.61 to 61.18% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 61.18%. The testing dataset AddOnSDK has f-measure in the range of 62.33 to 95.26% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 95.26%. The testing dataset Addons.mozilla.org has f-measure in the range of 63.89 to 90.12% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 90.12%. The testing dataset Webtools has f-measure in the range of 59.73 to 87.4% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 87.4%.

In case of SVM classifier the testing dataset Seamonkey has f-measure in the range of 76.61 to 95.42% for different training candidates. For this testing dataset Firefox as training candidate has given the maximum f-measure which is 95.42%. The testing dataset Thunderbird has f-measure in the range of 71.72 to 84.56% for different training candidates. For this testing dataset Seamonkey as training candidate has given the maximum f-measure which is 84.56%. The testing dataset Firefox has f-measure in the range of 77.85 to 91.46% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 91.46%. The testing dataset Boot2Gecko has f-measure in the range of 64.58 to 91.12% for different training candidates. For this testing dataset AddOnSDK as training candidate has given the maximum f-measure which is 91.12%. The testing dataset Bugzilla has f-measure in the range of 54.38 to 61.96% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 61.96%. The testing dataset AddOnSDK has f-measure in the range of 65.08 to 95.99% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 95.99%. The testing dataset Addons.mozilla.org has f-measure in the range of 68.57 to 90.41% for different training candidates. For this testing dataset Boot2Gecko as training candidate has given the maximum f-measure which is 90.41%. The testing dataset Webtools has f-measure in the range of 76.78 to 95.42% for different training candidates. For this testing dataset Thunderbird as training candidate has given the maximum f-measure which is 95.42%.

Figure 4 shows the maximum cross project f-measure of all testing datasets for the best training candidates in NB, KNN and SVM.

Results show that all the three classifiers gave same pattern of accuracy, recall and f-measure. We get maximum values of precision i.e. 100% for SVM across all datasets in comparison of KNN and NB.

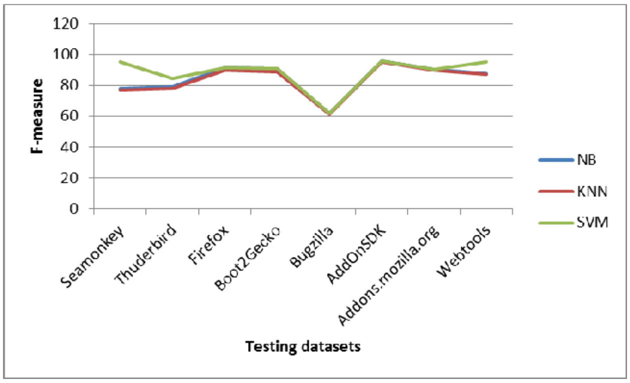

Fig. 4. Cross project f-measure (%) for best training candidates in NB, KNN and SVM

4 Related Work

An attempt has been made by Chaturvedi and Singh [5] to demonstrate the applicability of machine learning algorithms namely Naïve Bayes, k-Nearest Neighbor, Naïve Bayes Multinomial, Support Vector Machine, J48 and RIPPER to predict the bug severity level bug reports of NASA project. It was found that SVM works with significant accuracy level for open source and NB works with significant accuracy level for closed source projects. Another attempt has been made to predict the severity of reported bug by using summary attribute [1]. Researchers have studied the severity prediction using Naive Bayes classifier and categorized the bugs into severe and non-severe bugs. This study has been further extended to make a comparison with other data mining techniques like Support Vector Machine, Naive Bayes Multinomial and K-Nearest Neighbour for open source projects [3]. Further, a study has been done to determine the severity of reported bug for closed source bug repositories of NASA datasets using NB, MNB, SVM, KNN, J48, and RIPPER [4]. Results show that different machine learning techniques are applicable in determining the bug severity level 2, 3 and 4 with reasonable accuracy and f-measure. Our research has its roots in cross project prediction. Cross project priority prediction have been carried out by researchers and found that the priority prediction in cross project context is satisfactory with more than 70% accuracy and f-measure [6]. In another study authors used 12 real world applications and ran 622 cross-project defect predictions and found that only 21 predictions worked successfully with all precision, recall, and accuracy greater than 75% [7]. They showed that cross-project defect prediction is not symmetric. For example, data of Firefox can predict Internet Explorer defects well with precision 76.47% and recall 81.25% but do not work in opposite direction. They concluded that simply using historical data of projects in the same domain does not lead to good prediction results. They considered a similar problem cross-company defect prediction, using data from other companies to predict defects for local projects and analyzed 10 projects (7 NASA projects from 7 different companies and 3 projects from a Turkish software company SOFTLAB) [8].

They concluded that cross-company data increase the probability of defect detection at the cost of increasing false positive rate. They also concluded that defect predictors can be learned from small amount of local data.

In another study of cross-project defect prediction, 3 large-scale experiments on 34 datasets from 10 open source projects have been conducted [9]. In best cases, training data from other projects can provide better prediction results than training data from the same project. In 18 out of 34 cases recall and precision were greater than 70% and 50%. Another research was done to transfer learning for cross-company software defect prediction [10]. Recently, an attempt has been made to predict the severity levels of bugs in cross project context on a limited data set using SVM [11]. Authors used 7 datasets of Eclipse project to perform cross project severity prediction by using Support Vector Machine classifier. Out of total 210 cases only 36 cases show f-measure greater than 50%. This means the ratio of successful cross-project severity prediction is 17.14% which is very low. This show that selection of training data set should be done more carefully in the context of cross-project severity prediction.

It is clear from the literature that no effort has been made for cross project severity prediction by using bug attributes other than severity. Moreover, we have quantified the reported bug summary based on information gain criteria and derived bug complexity from bug fix time. In this paper, we have automated the prediction of multiclass bug severity and identification of best training candidate in cross project context by using multiple bug attributes.

5 Threats to Validity

Following are the factors that affect the validity of our study:

Construct Validity: The independent attributes taken in our study are not based on any empirical validation. Bug reports data in the software repository is highly imbalance due to which we are validating prediction results only for severity level 4.
Internal Validity: We have considered two derived attributes: complexity and summary weight. Values of these two attributes have not been validated against any predefined values as no predefined values are there in bug repositories.

External Validity: We have considered only open source Mozilla products. The study can be extended for other open source and closed source software.

Reliability: Statistica and RapidMiner have been used in this paper for model building and testing. The increasing use of these statistical software confirms the reliability of the experiments.

6 Conclusion

Bugs lying dormant in the software affect the quality and reliability of the software. A bug is characterized by many attributes. Different classifiers based on machine learning techniques help in predicting and assessing quantified values of these attributes. Earlier, authors have assessed the severity only on the basis of summary of a reported bug. In this paper, we have developed multiattribute based prediction models to assess the severity level of a reported bug in cross project context. The empirical results show that for SVM, KNN and NB classifiers, we get the same pattern of accuracy, recall and f-measure in cross project severity prediction using multiple attributes of a bug. For SVM we get maximum precision i.e. 100 % across all the datasets in comparison of KNN and NB. We found that multiattribute based cross project severity prediction provides accuracy, precision, recall and f-measure in the range of 37.34 to 91.63%, 94.99 to 100%, 44.88 to 97.86% and 61.18 to 95.99% respectively for different classifiers. The proposed work will help the projects whose historical data is not available to predict the severity of bug reports opened in new releases/development cycles. We found that the study to assess the severity level of a bug based on its different attributes will help bug triager in deciding the suitable developer and hence will improve the quality of the software product. In future we will extend our study for more machine learning techniques and datasets of open source and closed source software to empirically validate the study with more confidence.

References

1. Menzies, T., Marcus, A.: Automated severity assessment of software defect reports. In: IEEE Int. Conf. Software Maintenance, pp. 346–355 (2008)
2. Lamkanfi, A., Demeyer, S., Giger, E., Goethals, B.: Predicting the severity of a reported bug. In: Mining Software Repositories, MSR, pp. 1–10 (2010)
3. Lamkanfi, A., Demeyer, S.Q.D., Verdonck, T.: Comparing mining algorithms for predicting the severity of a reported bug. CSMR, 249–258 (2011)
4. Chaturvedi, K.K., Singh, V.B.: Determining bug severity using machine learning techniques. In: CSI-IEEE Int. Conf. Software Engineering (CONSEG), pp. 378–387 (2012)
5. Chaturvedi, K.K., Singh, V.B.: An empirical Comparison of Machine Learning Techniques in Predicting the Bug Severity of Open and Close Source Projects. Int. J. Open Source Software and Processes 4(2), 32–59 (2013)
6. Sharma, M., Bedi, P., Chaturvedi, K.K., Singh, V.B.: Predicting the Priority of a Reported Bug using Machine Learning Techniques and Cross Project Validation. In: IEEE Int. Conf. Intelligent Systems Design and Applications (ISDA), pp. 27–29 (2012)
7. Zimmermann, T., Nagappan, N., Gall, H., Giger, E.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 91–100 (2009)
8. Turhan, B., Menzies, T., Bener, A.: On the relative value of cross-company and within_company data for defect prediction. Empir. Software Engineering. 14(5), 540–578 (2009)
9. Zhimin, H., Fengdi, S., Ye, Y., Mingshu, L., Qing, W.: An investigation on the feasibility of cross-project defect prediction. Autom Software Engineering 19, 167–199 (2012)
10. Ma, Y., Luo, G., Zeng, X., Chen, A.: Transfer learning for cross-company software defect prediction. Information and Software Technology, Science Direct 54(3), 248–256 (2012)
11. Sharma, M., Chaturvedi, K.K., Singh, V.B.: Severity prediction of bug report in cross project context. In: Int. Conf. Reliability, Infocom Technologies and Optimization (ICRITO), pp. 96–102 (2013)
12. Sharma, M., Kumari, M., Singh, V.B.: Understanding the Meaning of Bug Attributes and Prediction Models. In: 5th IBM Collaborative Academia Research Exchange Workshop, I-CARE, Article No. 15. ACM (2013)