International Journal of Software Engineering and Its Application
Vol.12, No.1 (2018), pp.1-20
http://dx.doi.org/10.21742/ijseia.2018.12.1.01

A Review on Bug Report Assignment

Monika[1] and Nishi[2]

[1]*Research Scholar, DAV University, Jalandhar, India*
[2]*Assistant Professor, Dept. of CSE, DAV University, Jalandhar, India*
[1]*monika11.munjal@gmail.com*

Abstract

Nowadays, the software quality becomes an increasingly important issue. A huge number of bug reports by developers and users are received in bug tracking systems. Dealing with these bug reports in a manual way consumes time, resources and efforts. Numerous studies have been conducted using various techniques for the assignment of bug reports like information retrieval, classification, and clustering before classification that can overcome incorrect assignment. We review the bug assignment process because of their significance and popularity in an organized way. This paper provides a study for the bug reports to inspire the necessity for the work on bug report assignment and the existing work that has been performed on bug report assignment with possible problems that arises while working with bug assignment are summarized. Also, the practical analysis using various machine learning algorithm has been performed on the basis of a number of attributes and number of classes on Eclipse project containing 10,000 bug reports.

Keywords: *Bug report, Automatic Bug-Report Assignment, Machine learning techniques, Classification, Information retrieval*

1. Introduction

During software maintenance, bugs in software introduced due to various reasons which include poor designing, not properly understanding the requirements and series of changes in the coding after development. In software development, bug triaging is an essential software testing part and a vital task. It includes bug prioritization, detection of duplicate bug reports, assignment of a bug report. A bug repository plays an important role in handling bugs reports. These bugs are unavoidable and fixing them is expensive in development. There are two main issues related to bug data in software development tasks that may affect the bug repositories are a low quality and the large scale (Xuan et al. 2015). Some of the famous bug repositories are Bugzilla(www.bugzilla.org), JIRA(www.atlassian.com/software/jira), Trac(https://trac.edgewall.org) and *etc.* Examples of Eclipse and Mozilla which are famous datasets on bug reports which are used by many researchers and consists of information related to Eclipse and Mozilla respectively. Also, Software bugs are prevalent that they cost the U.S. economy around $59.5 billion annually, which is nearly 0.6 percent of the gross domestic product according to NIST (National Institute of Standards and Technology). A bug report in bug repository consists of several fields like “summary” provides the textual description of the bug, “severity” indicates the bug's importance, for example, trivial, enhancement *etc.*, priority marks the proper order in which the bug should be, a product like JDT, PDE *etc.*

When a user, developer or tester finds a bug in the software, a bug report is prepared and submitted to the bug repository. Then a defect manager makes a decision to whom the bug should be assigned *i.e.*, to the person who already has the experience in resolving the

Received (July 7, 2017), Review Result (January 3, 2018), Accepted (January 11, 2018)

ISSN: 1738-9984 IJSEIA

similar software bug. This process is referred as Bug Report assignment and has a great impact on the software quality if performed manually, assignment becomes labor-intensive, time- consuming and fault-prone resulting in bug tossing and delayed correction in a bug (Baysal *et al.*, 2009; Bhattacharya *et al.*, 2012). An empirical study by Jeong *et al.*, (2009) reports that the Eclipse project takes around forty days to assign a bug to primary developer and so it takes further hundred days or a lot of to depute to the second developer *i.e.*, nearly 25% of the bug reports are reassigned in Eclipse platform (Murphy *et al.*, 2011). Study on Eclipse project by Saha *et al.*, (2014) shows that there are a reasonable number of long-lived bugs and over 90 percent of them negatively affect the user's experience.

Various researchers have proposed making improvements to the challenges by automating the bug assignment based on classification using machine learning algorithms (Jeong *et al.*, 2009). A wide type of classifiers has been reported, and prior studies report prediction accuracy ranging from 40 to 60 %. (Anvik *et al.*, 2006; Jeong *et al.*, 2009; Ahsan *et al.*, 2009). Earlier work has targeted on open source software projects, principally the Eclipse and Mozilla projects. Very few studies in proprietary development projects on bug report assignment has done and that too on small organization (Helming *et al.*, 2011). Researchers have evaluated many machine learning algorithms for the purpose of classification. The two popular classifiers for assignment of bug reports are Support Vector MachineNaive Bayes and, explore by (Cubranic and Murphy 2004; Anvik *et al.*, 2006). Several other classifiers have been explored on bug assignment and compared prediction accuracy with various results (Ahsaan *et al.*, 2009; Helminh *et al.*, 2011; Bhattacharya *et al.*, 2012). For the improvement of the accuracy, some other authors have evaluated and presented customized solutions for bug report assignment for their specific projects (Xie *et al.*, 2012 and Xie *et al.*, 2013).

In this paper, we tend to specialize in the bug assignment task which has already received substantial awareness by the software development community. The main objectives of this review can be summarized as follows:

- To review the existing literature to evaluate the amount of work that has been published so far.
- To find out the significance and challenges faced by the researchers.
- To roll about the trends of research in bug report assignment area.
- To present the effects of automatic bug report assignment.

To the best of our knowledge, no work on bug assignment has been performed using associations among the bug features. These associations can support bug triggers to improve the prediction accuracy and process during development.

The rest of the paper is organized as follows. First, the Preliminary (terminologies and terms used in bug report assignment) of this research area is presented in Section 2. Then, the comparative study of the existing review and our review is presented in Section 3 and 4. In Section 5 we define significant results related to this research. Next, we discussed various gaps, conclusion and future work in Section 6 and 7 respectively.

2. Preliminary

This section presents some basic concepts of bug report assignment, the life cycle of bug reports, attributes in bug reports and the terms used in bug triage system.

2.1. Software Bug and Bug Report

A Software bug is an error or fault in a system which produces an unexpected result in the system (Nagwani *et al.*, 2011). The task of finding and removing a software bug is

called debugging. The cause of occurrence of bugs can be like mistakes made by a development team in collecting requirements, designing or tiny coding errors, which results in a very serious problem.

A bug report is a document which is consists of software bugs. It can be submitted by a developer, tester or by an end user. Mainly a bug report is consisting of bugId (identification number), product, Component, priority, severity (importance), summary, description, and title *etc*.

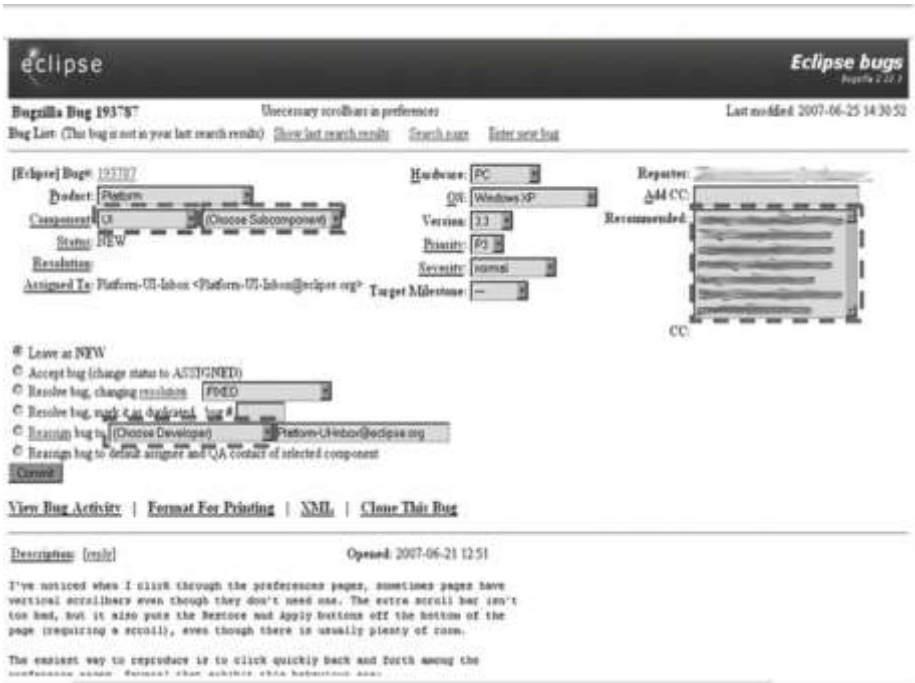


Figure 1. Bug Report (J. Uddin et al. 2016)

2.2. Bug Repository

Bug repositories are mostly consisting of open source projects like Eclipse where the developer, and end- user post the bug *i.e.*, problems with the system and comment on the problems related to the particular bug, which results in the improvement in the software quality. The widely-used bug repository is Bugzilla and it is visible to all people.

Table 1. Bug Repository: Bug Report Features and Class Label

Bug ID	Product	Comp	Severity	Priority	Summary
166107	Platform	UI	Normal	P3	[Quick Access] Quick Access should allov matching to an element for more than one string
167085	Platform	UI	Normal	P3	[QuickAccess] Quick Access shows null and duplicate entries
167486	Platform	UI	Normal	P3	[Quick Access] Remove Mnemnics from tooltip in ActionElement.java
168442	Platform	UI	Normal	P3	[Quick Access] Quick Access should no close when user tries to select a Node
433486	Platform	SWT	major	P3	Can't open FileDialog on OSX 10.9

2.3. Bug Attribute

A bug report contains attributes like Product, Component, Priority, Severity, Summary *etc*. Severity defines the impact of the bug. It can be divided into 7 parts Critical, Major,

Minor, Blocker, Normal, Trivial, Enhancement. Priority describes the order of bug in which that should be solved. In this P1 indicates the highest priority and P3 as lowest priority. The summary indicates the textual description of the bug. Table 1 depicts Some of the important bug attributes.

Table 2. Bug-report Attributes

Fields	Description
BugID	A unique number to identify the bug.
Summary	Mark the textual description of bug.
Severity	It indicates the importance of the bug. Example trivial, enhancement etc.
Priority	The order in which the bug should be.
Version	The version of the application in which the bug is found.
Resolution	Indicates what happens to this bug.
Status	Current state of the bug i.e. Resolved, New.
Product	Like JDT, PDE etc.
Date Created	When the bug was filed.
Date of close	When the bug was closed.
Platform orOS	Indicates the computing environment.
Component	Like Debug, TEXT etc.

2.4. Interactions with the Bug Report

There are different roles played by the persons in bug repository while interacting with the bug reports. The person who submits the report is known as a reporter or the submitter of the report. In this, Submitter is the person who submits the bug report. While the person who decides meaningfulness of the bug report is known as triager. Also, the resolver is the person who resolves the bug report. For a report, in which person may submit the report, assign the report to himself and contribute in the fix of bug (Anvik *et al.*, 2006), a single person behaves as a fulfilled all the roles.

2.5. Bug Triaging System

Bug report triage can be categorized as (a) Find out whether the bug is a real bug or not, (b) Checking for the duplication of the bug report, (c) Prioritizing the bug report, (d) and assigning the appropriate developer to resolve the bug report submitted. (Cubranic *et al.*, 2004).

2.6. Why bug report assignment?

In Software maintenance, Bug report assignment is a crucial part. In particular, wrong assignment of the bug reports to the developers can be very costly in large software development projects. Several studies propose automating bug assignment using machine learning in open source software projects. But still no study exists for a large scale proprietary related projects in industry. (Jonsson *et al.*, 2015) And after the thorough review, we found that work on bug report assignment mostly uses the techniques like machine learning, information retrieval, and some of mathematical theories like fuzzy sets *etc.*, (Zhang *et al.*, 2015).

2.7. The Life Cycle of Bug Report

A bug report goes through several resolution statuses over its lifetime. Figure 2 delineates the life cycle of a bug report. When a bug report is first submitted, it is marked as UNCONFIRMED. Then a triager verifies the duplication of the bug report submitted,

when a bug is not duplicate to the existing bug, then the status is changed to SET. And the triager assigns that bug report to the appropriate developer and the status is now Changed to ASSIGN. Then the developer reproduces the bug and tries to resolve it. When that is solved by that developer the status is changed to RESOLVE.

Now, if the tester is not satisfied with the resolves bug, the bug should again reopen with status as REOPEN. And if tested is satisfied and verified with the solution then the status is set to VERIFIED. Then in the last phase status is set to CLOSED, when there is no occurrence of the bug report. Note if the bug is solved by making the necessary change in code the status is set from RESOLVED to FIXED. And if the bug is not solved due to some reason then the set changed to WONTFIX. (Zhang *et al.*, 2015).

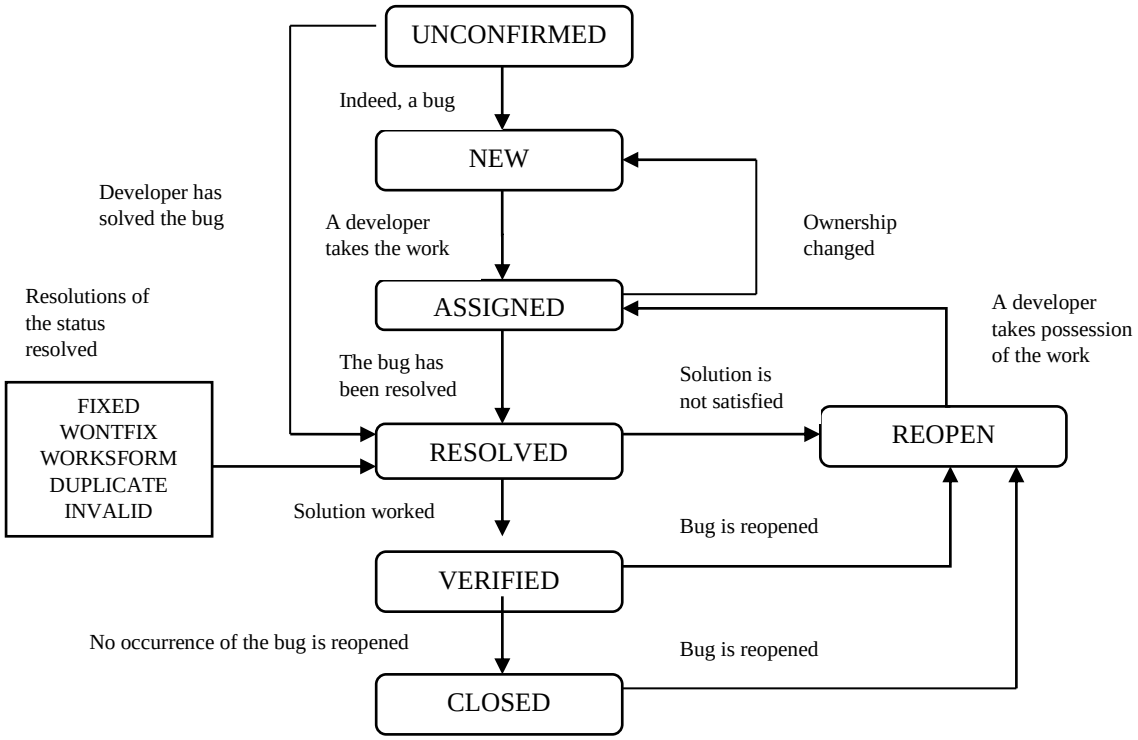


Figure 2. The Life Cycle of Bug Report (J. Uddin et al. 2016)

3. Literature Review

This section presents several relevant papers to describe the significant machine learning algorithms for bug report assignment. For our survey, six search venues are targeted (70 Papers). They are:

- IEEE Xplore (Institute of Electrical and Electronics Engineers)
- Science Direct.
- ACM Digital Library (Association for Computing Machinery).
- Springer.
- Wiley (Wiley Online Library)
- IET (The Institute of Engineering and Technology)
- Also, the related papers which were not available on these venues are taken from Google Scholar and come under “others” category.

Search terms used in our survey are:

- Bug Assignment
- Bug report Assignment
- Bug Assignment Prediction

- Prediction of Bug report assignment.
- Bug Triaging

The key idea about each paper is discussed below:

Cubranic and Murphy (2004) were the first to propose the idea of machine learning to semi-automate bug report assignment approach. They used developer's id and terms of title and description of the bug report as features on 15,859 bug reports. Trained Naive Bayes Classifier (NB) and achieved 30% accuracy.

Furthermore, Podgurski *et al.*, (2003) they evaluate the prioritizing and classifying variety of software faults using machine learning techniques. Canfora and Cerulo (2006) applied a probabilistic information retrieval based bug assignment and report 50% recall.

Later, Anvik *et al.*, (2006) used three machine learning classifiers, Naive Bayes (NB), Support Vector Machine (SVM), and C4.5 along with the concept of filters *i.e.*, by filtering out "invalid" or "wont fix" bug reports and developers who fixed less than nine bugs or no longer contribute to the project. They observed SVM performs better than others with 64% accuracy.

To improve the accuracy of bug triage, Anvik (2007) extended pioneering work by using rule-based classification and Expectation Maximization (EM). Also, discussed repository-oriented and development-oriented decision for automating bug report assignment. They achieved 70% accuracy on Eclipse by using 3,338 bug reports and also used four more datasets for the evaluation of their framework. Anvik andMurphy (2011) used Nearest Neighbor (NN) classifier. Several other researchers continued by Anvik from different platforms or datasets using various classifiers on bug report assignment.

To automatically assign a developer to the new bug report, Ahsan *et al.*, (2009) used Decision trees, Random Forest, REPTree and RBF Network with product, component, platform as features.

Instead of focusing on the open source project Lin *et al.*, (2009) experimented machine learning bug assignment on Chinese proprietary SoftPM project. They used 2,576 bug reports and report 77.64% average accuracy by taking the module of the bug, component, bug type, bug class, phase id, priority, and submitter.

Matter *et al.*, (2009) trained a model by extracting vocabulary from developer's source code. They implemented approach on 130,769 bug reports of Eclipse platform and reported an accuracy of 33.6% of top 1 developers and 71% of top 10 developers. Bhattacharya and Neamtiu (2010) proposed a refined classifier with tossing graph and feature vector which improved the accuracy of bug report triage.

Table 3. Year Wise Summary of Filtered Papers

Year	No. of Filtered papers
1999	1
2003	1
2004	1
2006	2
2008	1
2009	5
2010	3
2011	5
2012	7
2013	6
2014	11
2015	12
2016	7

Xuan *et al.*, (2010) worked on semi-supervised approach by using both Naive Bayes classifier (supervised classification) and Expectation Maximization (unsupervised classification) together, which generates weighted recommendation list. They evaluated that their approach avoids low-quality bugs and improved the accuracy of bug triage by 6%.

Park *et al.*, (2011) proposed an algorithm of cost-aware bug triage. They trained SVM classifier with bug features and reduced 30% cost of bug assignment. Nagwani *et al.*, (2011) in their proposed work used the textual information of the bug's summary and description. They generated the frequent terms of information extracted and for the prediction of the appropriate developer, the concept of term similarity was used.

Extending their prior work, Bhattacharya *et al.*, (2012) addressed a probabilistic bug tossing graph based on multiple dimensions *i.e.*, feature selection (Product, Component), a classifier (Naive Bayes) and incremental learning. They examined their work on Eclipse and Mozilla, with 85,6259 bug reports. They achieved average accuracy 77.87% for Mozilla and 77.43% for Eclipse using Naive Bayes classifier.

Kanwal *et al.*, (2012) evaluated the comparison of Naive Bayes and Support vector machine as well as the features (attributes) that report better results for bug prioritization. They validate their proposed work by the recall, precision, Nearest false negatives and Nearest false positives. Dommati *et al.*, (2012) performed feature selection and extraction of bug data. For their evaluation, they used accuracy and recall measures. Chaturvedi *et al.*, (2012) worked on bug severity task, using rule-based classifier for binary classes (severe or non- severe) in the bug report.

Recently, Tian *et al.*, (2013) in their proposed work handled the imbalance of bug reports using different machine learning classifier, which recommends priority of the bug report. They worked on 100,000 bug reports of Eclipse and validated using the F-measure parameter. Hao *et al.*, (2014) proposed a new tokenization algorithm and used VSM (Vector Space Model) for extracting bug similarities. They implemented on large projects which result in better accuracy.

Punitha and Chitra (2013) presented a critique of software defect Prediction using testing metrics. In their survey, they identified defective modules and the main goal of their research is helping the software developers in identifying defects using data mining techniques and software metrics.

A survey on data mining in software engineering using different clustering techniques was conducted by Kaur and Garg (2014). In their work, they explained about the different clustering techniques used for different tasks of coding, testing, maintenance, debugging. They provide the discussion and conclusion that every technique has their own advantage and disadvantage and no particular technique is best for all tasks.

Aggarwal *et al.*, (2014) presented the various applications of topic modeling for multi-faceted corpus exploration and analysis of mining issue tracking system. They studied different fields like bug status, operating system, time category, priority and bug type.

Begel *et al.*, (2014) conducted two surveys to understand the questions asked for data scientist by software engineers about software. The first survey solicited for the investigation of the question by data scientist about software engineers, processes and practices and second survey for the ranking of the questions.

Survey on Types of bug reports and classification techniques in data mining was presented by Mishra and Kumar (2015). In their study, they discussed a how the different types of classification techniques used for the dealing of bug reports.

Xia *et al.*, (2015) developed imbalanced ML.KNN model consists of different features (textual, meta and mixed) and three multilabel classifiers for bug report field reassignment. They performed their proposed work on four open source projects with 190,558 bug reports and used precision, recall and F-measure for evaluation metrics.

Yang *et al.*, (2015) worked on semi-automatic bug severity prediction using topic model and multi-feature approach. Using bug repositories, they extracted the topics from

historical bug reports and used features like product, component, priority, severity etc.(multi-features) to find bug report related to each topic. For a new bug report, they recommend the corresponding developer with similar multi-features for prediction of severity. They not only evaluated their work on 30,000 bug reports but also, compared related work on bug severity.

Zhang *et al.*, (2015) presented some empirical study on the background of bug reports analysis on Bugzilla. They summarized the necessity of bug report analysis, a survey of existing work on the bug report and some possible problems which need the attention for research on bug reports.

Most recently, Jonsson *et al.*, (2015) in their proposed work used the concept of Stack Generalization which combines different classifiers (Bayes net classifier, Naive Bayes classifier, SVM, KNN, Decision tree classifier *etc*.,) and implemented their work on five different software projects. They also presented the relative value of textual and non-textual features.

Anvik *et al.*, (2015) proposed a new Time-tf-idf term weighting technique (that uses the time when the developer used the terms during text analysis process) for the improvement of automatic bug assignment and compared the accuracy and MRR of Time-tf-idf with other machine learning algorithms. Xuan *et al.*, (2015) combined the concept of feature selection (CH) and instance selection (ICF) to simultaneously reduce the data scales as well as to improve the data quality. They evaluated that ICF -> CH based on Naive Bayes classifier obtains better results than KNN and SVM.

From the above review, we observe that many studies about bug assignment receive the attention of the researchers using different machine learning algorithms as shown in Figure 1(a), but no work has been done on Bug Assignment using associations between nominal features. The accuracy of a classifier highly dependent on the feature vector in the training dataset. Most of the existing techniques used different nominal and textual features as shown in Figure 1(c) for bug assignment but still a combination of product, component, priority, severity, and summary is not used. Accuracy is the big concern in the case of large number of reports as shown in the Figure 1(b), as the number of reports increases accuracy decreases, only Bhattacharya *et al.*, (2012) succeeded to achieve high accuracy with large number of reports and further he improved by Bhattacharya by applying graph tossing techniques with machine learning algorithms.

Table 4. Overview of Classification and Information Retrieval Techniques

Researcher	Classification Technique	Information retrieval	Features used	Datasets
Cubranic and Murphy (2004)	Naïve Bayes	-	Title, Description	Eclipse
Canfora and Cerulo (2006)	-	Probabilistic	Title, Description, Comments	Mozilla, KDE
Anvik et al. (2006)	Naïve Bayes, **SVM, C4.5**	-	Title, Description	Eclipse, Firefox, gcc
Baysal et al. (2009)	-	RSSE	Title, Description	Not evaluated
Matter et al. (2009)	-	RSSE	Description	Eclipse
Ahsan et al. (2009)	Decision trees, Naïve Bayes, RBF Network, Random forest, REPTree, **SVM**, C4.5	-	Product, Component, Platform, Title	Mozilla
Lin et al. (2009)	SVM, **C4.5**	-	Title, Description, Component, Type, Phase, Priority, Submitter	SoftPM

Jeong et al. (2009)	Naïve Bayes, Bayesian Network	-	Title, Description	Eclipse, Mozilla
Xuan et al. (2010)	Naïve Bayes	-	Title, Description	Eclipse
Aljarah et al. (2011)	Bayesian Networks	-	Title	Eclipse
Helming et al. (2011)	Decision trees, Naïve Bayes, Nearest Neighbor, Neural Networks, **SVM,** Constant classifier	-	Title, Description	King's Tale, UNICASE, DOLLI
Park et al. (2011)	-	RSSE	Description, Platform, Version, Phase	Apache, Eclipse, Linux Kernel, Mozilla
Anvik and Murphy (2011)	Naïve Bayes, Nearest Neighbor, **SVM,** Rules, Expectation MaximizationC4.5	-	Title, Description	Eclipse, Firefox, gcc, My Lyn, Bugzilla
Chen et al. (2011)	-	Algebraic	Title, Description	Eclipse, Mozilla
Bhattacharya et al. (2012)	**Naïve Bayes**, Bayesian Networks, SVM, C4.5	-	Product, Component	Eclipse, Mozilla
Kagdi et al. (2012)	-	Algebraic	Comments, Identifiers	ArgoUML, Eclipse, Koffice jEdit,
Linares-Vasquez et al. (2012)	SVM	**Algebraic**	Title, Description, Comments, Identifiers	ArgoUML, muCommander
Nagwani and Verma (2012)	-	Algebraic	Title, Description	Mozilla
Shokripour et al. (2012)	-	Algebraic	Comments	Eclipse, Mozilla, GNOME
Alenezi et al. (2013)	Naive Bayes	-	Title	Eclipse, NetBeans, Maemo
Jonsson et al. (2015)	Decision trees, Naïve Bayes, Bayesian Networks, Random forest, REPTree, **SVM,** Rules	-	Title, Description, Version, Type, Priority, Submitter	Telecom, Automation

Table 5. Overview of Bug Reports and Evaluations Performed in Previous Studies

Researcher	Datasets	Bug reports	Accuracy	Other Evaluation
Cubranic and Murphy (2004)	Eclipse	Eclipse (15,859)	0.3	-
Canfora and Cerulo (2006)	Mozilla, KDE	Mozilla (12,477), KDE (14,395)	0.32, 0.59	-
Anvik et al. (2006)	Eclipse, Firefox, gcc	Eclipse (8,655), Firefox (9,752), gcc (2,629)	0.58, 0.64	F@ 1:0.006
Matter et al. (2009)	Eclipse	Eclipse (130,769)	-	F@ 1:0.3
Ahsan et al.	Mozilla	Mozilla (1,983)	0.44	-

(2009)				
Lin et al. (2009)	SoftPM	SoftPM (2,576)	0.78	-
Jeong et al. (2009)	Eclipse, Mozilla	Eclipse (211,822), Mozilla (429,903)	-	Rc@ 2:0.58, 0.56
Xuan et al. (2010)	Eclipse	Eclipse (20,000)	0.2	-
Aljarah et al. (2011)	Eclipse	Eclipse (38,843)	-	F@ 1:0.57
Helming et al. (2011)	King's Tale, UNICASE, DOLLI	King's Tale (256), UNICASE (1,191), DOLLI (411)	0.40, 0.30, 0.40	-
Park et al. (2011)	Apache, Eclipse, Linux Kernel, Mozilla	Apache (656), Eclipse (47,862), Linux Kernel (968), Mozilla (48,424)	0.70, 0.40, 0.30, 0.65	-
Anvik and Murphy (2011)	Eclipse, Firefox, gcc, My Lyn, Bugzilla	Eclipse (6,500), Firefox (3,400), gcc (2,600), My Lyn (700), Bugzilla (850)	-	F@ 1:022, 0.02, 0.06, 0.46, 0.10
Chen et al. (2011)	Eclipse, Mozilla	Eclipse (115,058), Mozilla (119, 852)	-	-
Bhattacharya et al. (2012)	Eclipse, Mozilla	Eclipse (306,296), Mozilla (550,000)	0.70, 0.70	-
Linares-Vasquez et al. (2012)	jEdit, ArgoUML, muCommander	jEdit (200), ArgoUML (100), muCommander(100)	-	F@ 1:0.05, 0.31, 0.60
Nagwani and Verma (2012)	Mozilla	Mozilla	-	Qualitative
Shokripour et al. (2012)	Eclipse, Mozilla, GNOME	Eclipse (35,140), Mozilla (9,917), GNOME (119,176)	0.31, 0.27, 0.28	-
Alenezi et al. (2013)	Eclipse, NetBeans, Maemo	Eclipse (7,561,6,791), NetBeans (11,311), Maemo (3,505)	-	F@ 1:034, 0.35, 0.20, 0.48
Jonsson et al. (2015)	Telecom, Automation	Telecom (35,000), Automation (15, 113)	0.71,0.57, 0.87, 0.79, 0.50	-

4. Experiments

Experimental Setup: We used Eclipse bugs to measure the accuracy of our existing and proposed algorithm. We analyzed the entire lifespan of application. For, eclipse we consider bug numbers from 1 to 10,000. Eclipse bug reports have been found to be high quality, which helps reduce noise when training the classifiers.

Data collection: We used the bug reports to collect three kinds of data:

1. Keywords: We collect keywords from the bug title, bug description, and comments in the bug report.

2. Bug Source: We retrieve the product, component, priority, severity, and summary the bug has been filed under from the bug reports.

3. Temporal information: We collect information about when the bug has been reported and when it has been fixed.

4.1. Classification based on Number of Attributes

4.1.1. Bug Assignment Classification Results for SVM (Support Vector Machine): It is an algorithm that determines non-linear models describing each class and then uses these models to predict the class of new instances. From the point of view of the support vector machine algorithm, the different instances exist in a multi-dimensional space defined by the attributes. For example, if the instance has four attributes, then the instances are placed in a four-dimensional space according to their attribute values.

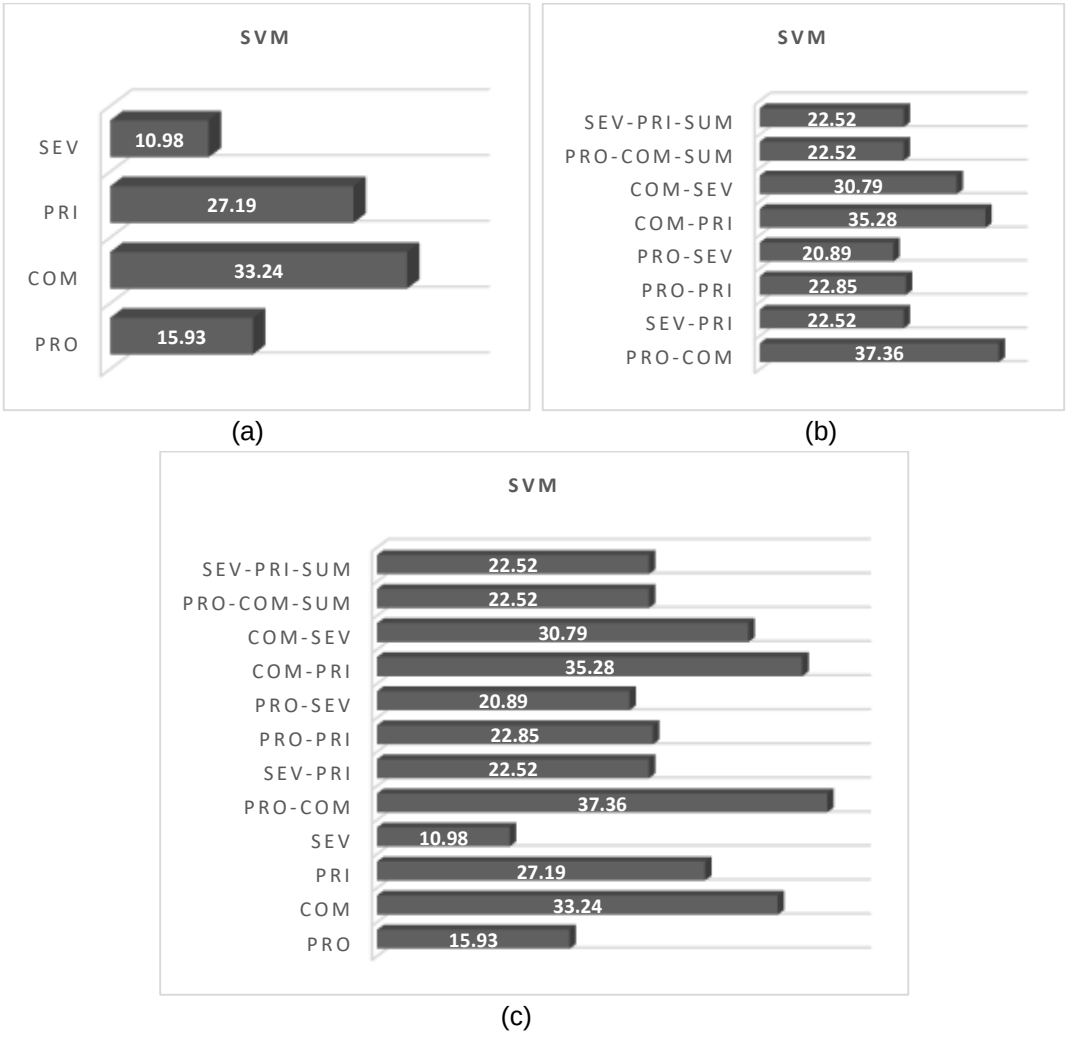


Figure 3. (a)Classification using Single Attribute. (b) Classification using Two or More Attributes. (c) Classification based on SVM

4.1.2 Bug Assignment Classification Results for Recursive Partitioning and Regression Trees: The rpart program builds classification or regression models of a

general structure using a two-stage procedure; the resulting models can be represented as binary trees. The tree is built by the following process:

a) First, the single variable is found which best splits the data into two groups. The data is separated and then this process is applied separately to each sub-group and so on recursively until the sub-groups either reach the minimum size or until no improvement can be made.

b) The second stage of the procedure consists of using cross-validation to trim back the full tree.

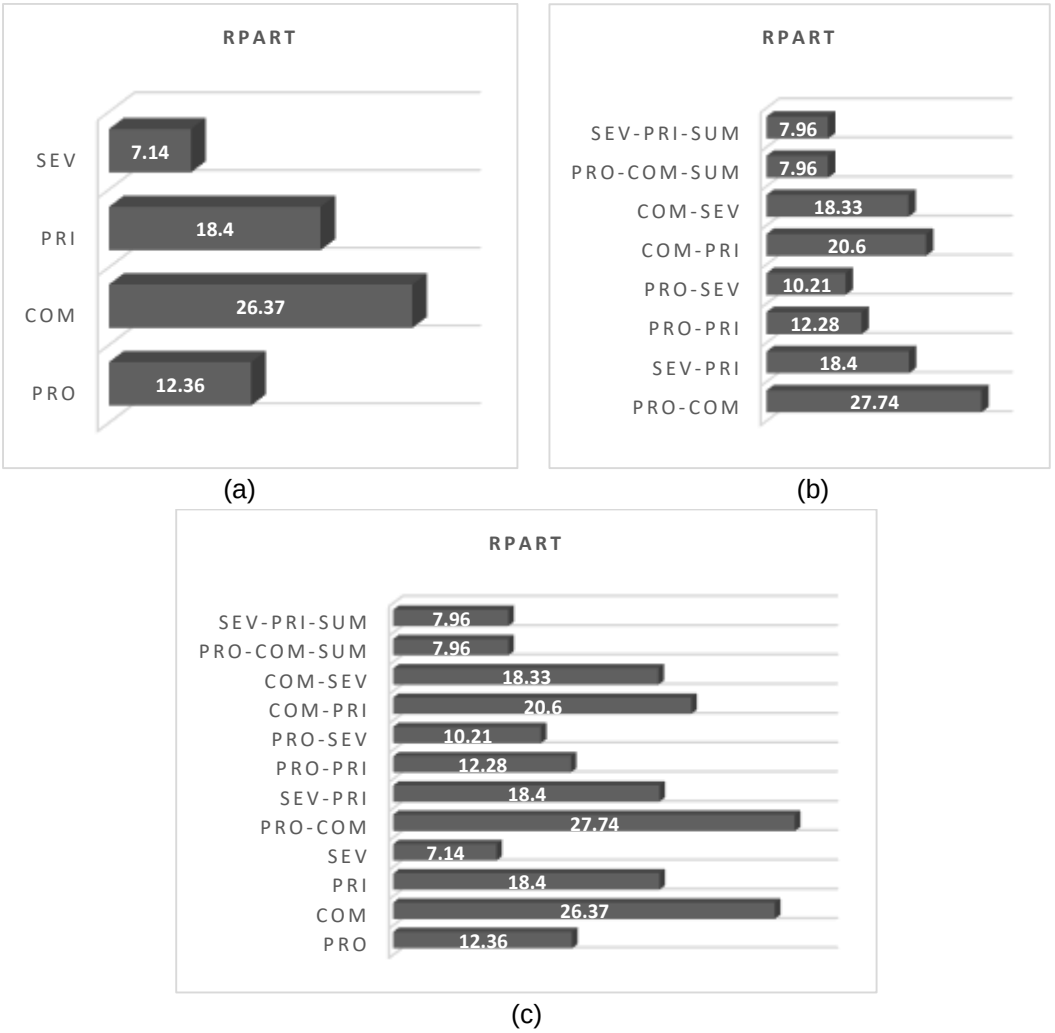


Figure 4. (a)Classification using Single Attribute. (b) Classification of two or More Attributes. (c) Classification based on rpart

4.1.3. Bug Assignment Classification Results for C5.0: C5.0 algorithm is used to build either a decision tree or a rule set. This model works by splitting the sample based on the field that provides the maximum information gain. Each sub sample defined by the first split is then split again, usually based on different fields., and the process repeats until the subsamples cannot be split any further. Finally, the lowest level splits are reexamined, and those that don't contribute significantly to the value of the model are removed and pruned.

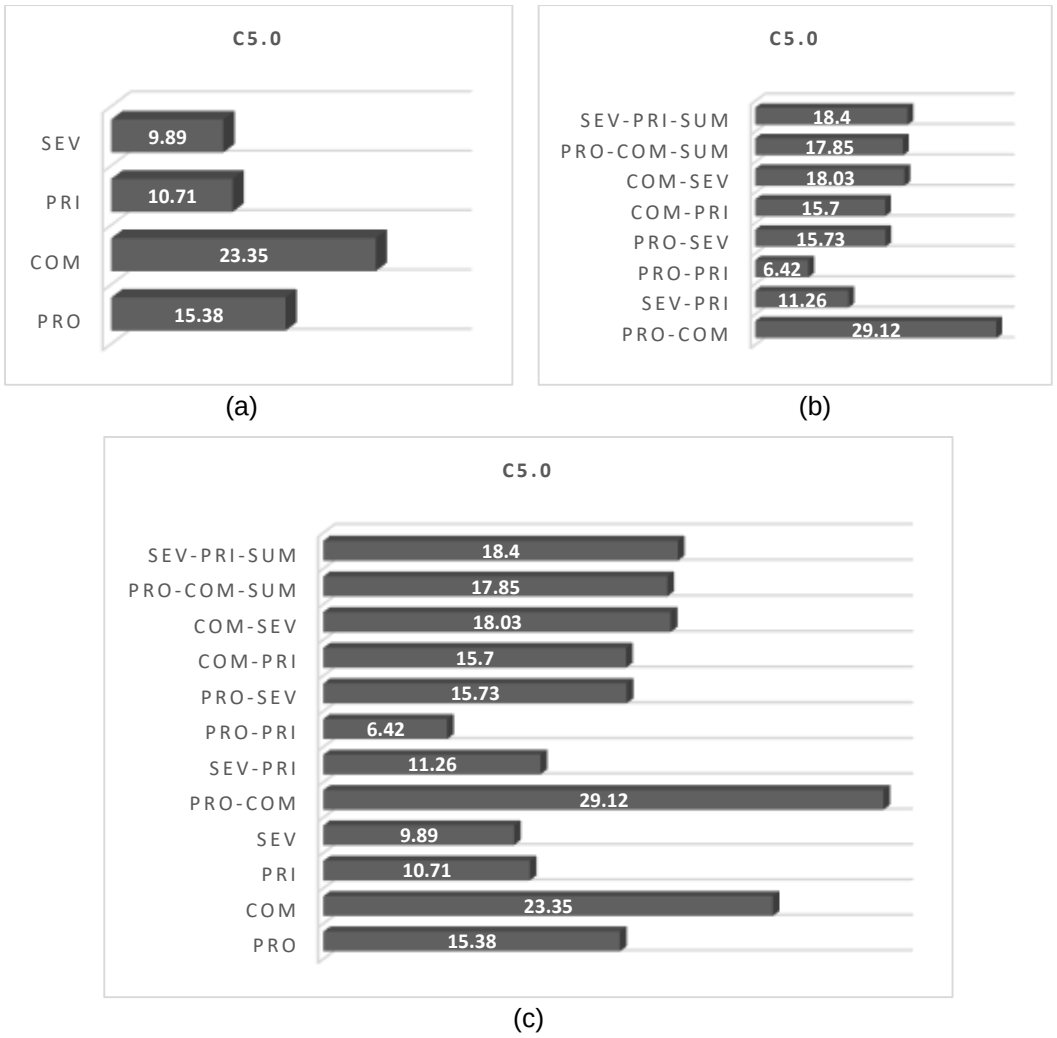


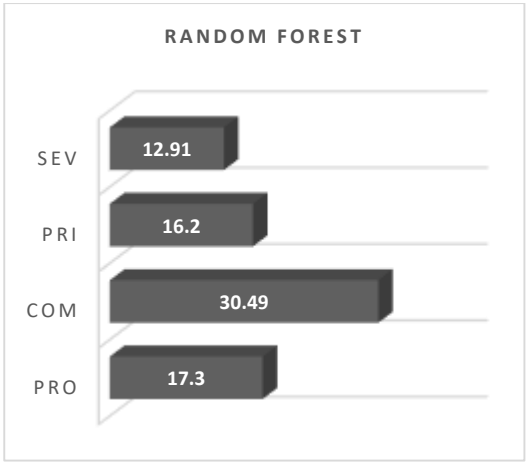
Figure 5. (a) Classification of Single Attribute. (b) Classification of two or more Attributes. (c) Classification based on C5.0.

4.1.4 Bug Assignment Classification Results for Random Forest: Random forest is the term for an ensemble of decision trees. To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is planted as follows:

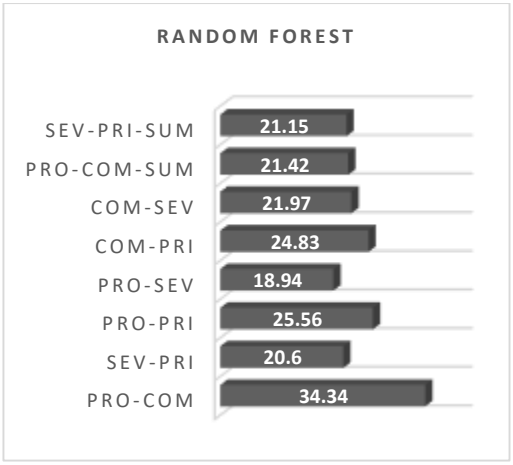
a) If the number of cases in the training dataset is N, the sample of N cases is taken at random but with replacement. This sample will be a training set for growing the tree.

b) If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split of this m is used to split the node. The value of m is held constant during the forest growing.

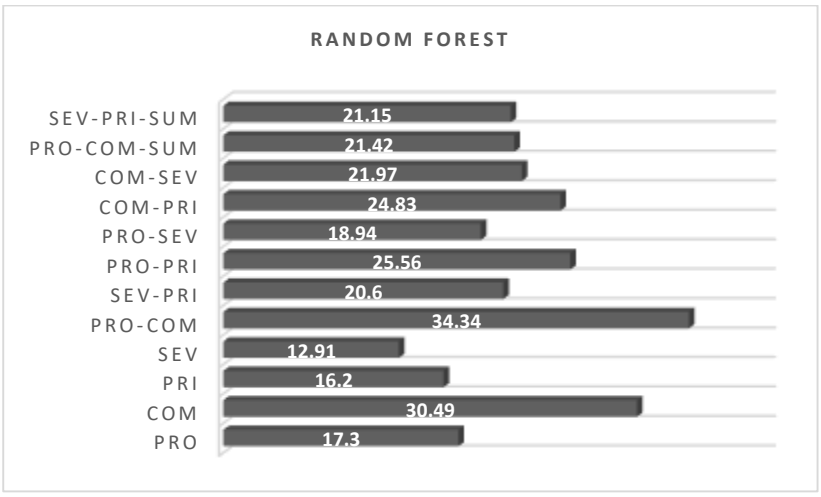
c) Each tree is grown to the largest extent possible. There is no pruning.



(a)



(b)



(c)

Figure 6. (a) Classification of Single Attribute. (b)Classification of Two or More Attributes. (c) Classification based on Random Forest

Conclusion: We demonstrate that, the highest prediction accuracy is achieved using combination of two attributes i.e. Product and Component with SVM Classifier as shown below:

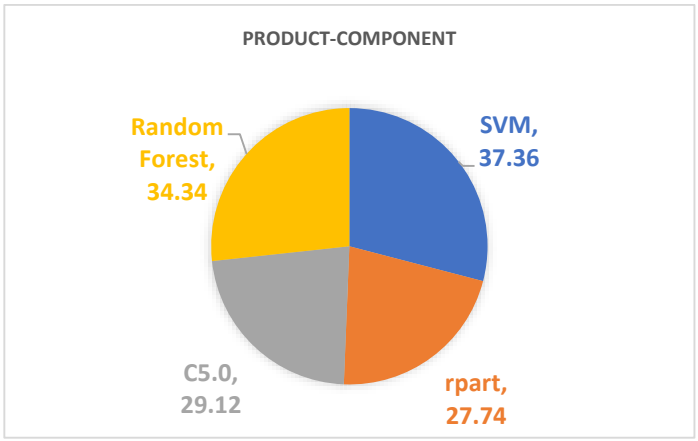


Figure 7. Product-Component Attributes with Different Algorithms

4.2. Classification based on Number of Assignee

Table 6. Individual Classifier Results for Bug Assignment Prediction Accuracy

	Accuracy			
Assignee	SVM	C5.0	Rpart	Random Forest
top5	44.37	40.44	32.37	42.6
top10	38.79	38.79	28.15	37.21
top15	36.18	29.73	26.68	36.18
top 20	34.06	28.76	24.66	35.53
top25	32.43	27.80	23.50	33.00
top30	31.43	26.99	22.67	31.91

Conclusion: We demonstrate that the highest prediction accuracy is achieved using SVM Classifier with top 5 assignees in the Eclipse dataset.

4.3. Classification Techniques on Eclipse Dataset with 10,000 Bug Reports Consists of 134 Assignees and 5 Attributes

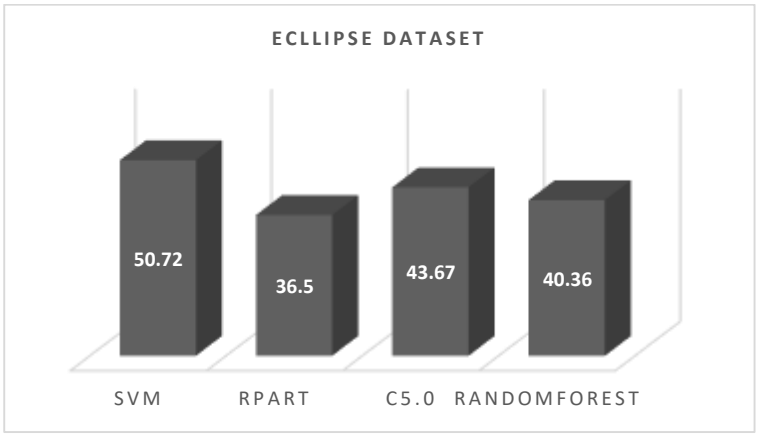


Figure 8. Comparison of Accuracy of Various Algorithm

Conclusion: We demonstrate that the highest prediction accuracy is achieved using SVM Classifier with 50.72% accuracy, while C5.0 gives the accuracy 43.67%, accuracy contained by Random forest 40.36% and rpart gives the accuracy of 36.5%. This accuracy can be increased by using classification based on association rules.

5. Significant Results

In this section, we define the significant results that were found in this review study. It consists of significant techniques used, evaluation metrics, the dataset used, researchers and significant venues.

5.1. Significant Techniques Used

To resolve the issue of assignment of bug reports researchers mainly used the classification techniques and most well-known techniques used were SVM, Naïve Bayes, and C4.5. Researchers also used Bayesian NETworks, Decision trees, Nearest Neighbor, Neural Network, RBF Network, Random Forest, REPTree, Rules, Expectation Maximization and Constant Classifier.

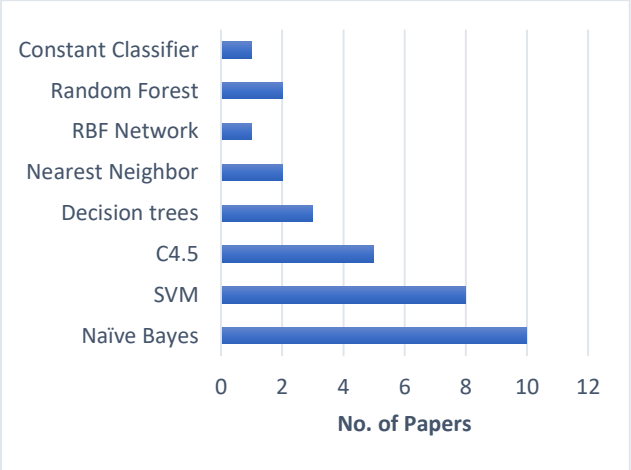


Figure 9. Significant Techniques used5.2 Significant Dataset Used

In Figure, the datasets commonly used by various researchers for the assignment of bug reports are shown. It clearly shows that the mostly used datasets by researchers were Eclipse and Mozilla repositories. Researchers also used gcc, Firefox, NetBeans, KDE, GNOME etc. It can be seen that almost all researchers used a dataset which from the open source repositories.

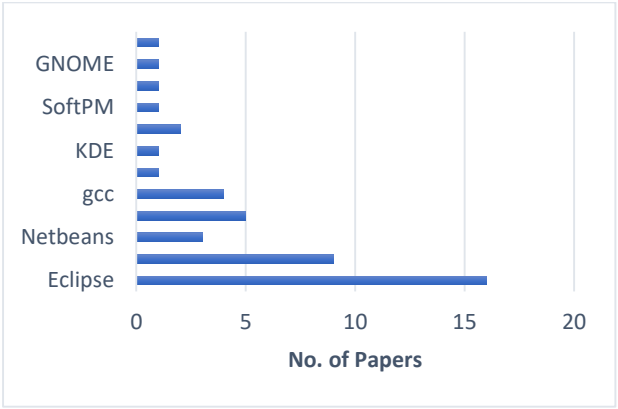


Figure 10. Significant Datasets Used.5.3 Significant Evaluation Metrics Used

The commonly used evaluation metrics by researchers are Precision, recall, F-measures, and Accuracy.

5.4. Significant Venues

According to our survey, the publications on bug report assignment have been mostly published by Elsevier and Springer. Also, related work on bug assignment like bug prioritization, bug severity prediction *etc.*, which is part of bug triage have most of the papers published by IEEE and ACM only. Remaining work on bug assignment or bug report reassignment that published other venues are IEEE, conference etc. are included in others.

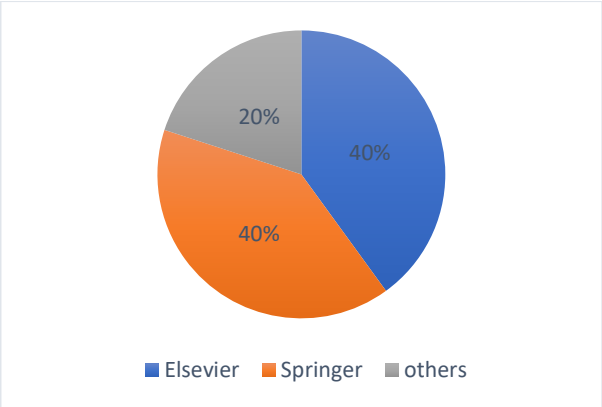


Figure 11. Significant Venue

6. Discussion Gaps

Many studies about bug report assignment receive the awareness of the researchers. However, there are still missing gaps that need to be filled. We summarize some of the gaps in bug report assignment process as follows:

1. Many researchers have worked on automated bug-report triage using various machine learning algorithms. The main issues with traditional machine learning algorithms are these algorithms require a large amount of labeled data for the training of the classifier. (Nigam *et al.*, 2012).

2. We observe that many studies about bug assignment receive the attention of the researchers using different machine learning algorithms but no work has been done on Bug Assignment using associations between nominal features.

3. Also, classifiers performance can be increased when it as applied to the clustered dataset for bug report assignment as they have proven to be promising for bug prioritization (Goyal *et al.*, 2015) but this approach only considered a single feature for validation.

4. Accuracy is the big concern in the case of a large number of reports as shown in the as the number of reports increases accuracy decreases.

5. One cannot afford to assign an incorrect developer to the bug reports. However, with proposed machine learning algorithms have provided good results but still, performance can be improved by in terms of accuracy, recall, precision, and F-measures.

6. The accuracy of a classifier highly dependent on the feature vector in the training dataset. Most of the existing techniques used different nominal and textual features for bug assignment but still a combination of product, component, priority, severity, and summary is not used.

7. Conclusion and Future Work

When a developer or user submits a new bug report in bug tracking system, the trigger makes decisions about several fields of the bug report such as component, product, assignee, severity, priority levels etc. Handing bug reports manually is often results in time and resource consuming task. There is a reasonable volume of research on bug report assignments.

This paper summaries the existing literature review which covers a variety of work in the area of the bug triaging that mainly deals with bug report assignment. In the Review, we first briefly discussed some preliminaries for bug report assignment. Secondly,

through literature survey method some statistics on bug assignment research to show the amount of work on it was presented till now. Third, we graphically showed the results of analysis experimented by us on Eclipse project with 10,000 bug reports. Fourth, different types of significant outcomes of our review were presented. We finally discussed some discussion gaps that should be completed for future research.

This study focus on bug report assignment, however, our future plan is to explore new algorithms and strategies to improve the bug triaging system. Also, the research gaps presented in the study needs to enhance the performance of well-known classification algorithms.

References

[1] V. Akila, G. Zayaraz and V. Govindasamy, "Effective Bug Triage–A Framework", Procedia Computer Science, vol. 48(Iccc), **(2015)**, pp. 114-120, https://doi.org/10.1016/j.procs.2015.04.159.
[2] J. Anvik, L. Hiew and G. C. Murphy, "Who should fix this bug?", Proceeding of the 28th International Conference on Software Engineering - ICSE '06, **(2006)**, pp. 361, https://doi.org/10.1145/1134285.1134336.
[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage", ACM Transactions on Software Engineering and Methodology, vol. 20, no. 3, pp. 1-35, https://doi.org/10.1145/2000791.2000794.
[4] P. Bhattacharya, I. Neamtiu and C. R. Shelton, "Automated, highly-accurate, bug assignment using machine learning and tossing graphs", Journal of Systems and Software, vol. 85, no. 10, **(2012)**, pp. 2275-2292, https://doi.org/10.1016/j.jss.2012.04.053.
[5] K. K. Chaturvedi and V. B. Singh, "Determining Bug severity using machine learning techniques", 2012 CSI 6th International Conference on Software Engineering, CONSEG 2012, (C), **(2012)**, pp. 2005-2009, https://doi.org/10.1109/CONSEG.2012.6349519.
[6] A. E. Hassan and T. Xie, "Software intelligence: The Future of Mining Software Engineering Data", Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research - FoSER '10, **(2010)**, pp. 161, https://doi.org/10.1145/1882362.1882397.
[7] A. Hindle, A. Alipour and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection and ranking", Empirical Software Engineering, vol. 21, no. 2, **(2016)**, pp. 368-410, https://doi.org/10.1007/s10664-015-9387-3.
[8] W. Husain, P. V., Low, L. K. Ng and Z. L. Ong, "Application of Data Mining Techniques for Improving Software Engineering", **(2011)**, pp. 1-5.
[9] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh and P. Runeson, "Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts", Empirical Software Engineering, vol. 21, **(2016)**, https://doi.org/10.1007/s10664-015-9401-9.
[10] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage", Journal of Computer Science and Technology, vol. 27, no. 2, **(2012)**, pp. 397-412, https://doi.org/10.1007/s11390-012-1230-3.
[11] M. Kaur and S. K. Garg, "Survey on Clustering Techniques in Data Mining for Software Engineering", **(2014)** May.
[12] N. K. Nagwani and S. Verma, "Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes", International Conference on ICT and Knowledge Engineering, **(2011)**, pp. 113-117, https://doi.org/10.1109/ICTKE.2012.6152388.
[13] J. woo Park, M. W. Lee, J. Kim, S. won Hwang and S. Kim, "Cost-aware triage ranking algorithms for bug reporting systems", Knowledge and Information Systems, vol. 48, no. 3, **(2016)**, pp. 679-705, https://doi.org/10.1007/s10115-015-0893-9.
[14] R. Raman, "Advances in Intelligent Informatics", Advances in Intelligent Systems and Computing, vol. 320(C), **(2015)**, pp. 621-631, https://doi.org/10.1007/978-3-319-11218-3.
[15] S. C. Satapathy, B. N. Biswal, S. K. Udgata and J. K. Mandal, "Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014", Advances in Intelligent Systems and Computing, vol. 327, **(2014)**, pp. 387-395, https://doi.org/10.1007/978-3-319-11933-5.
[16] G. Sharma, S. Sharma and S. Gujral, "A Novel Way of Assessing Software Bug Severity Using Dictionary of Critical Terms", Procedia Computer Science, vol. 70, **(2015)**, pp. 632-639, https://doi.org/10.1016/j.procs.2015.10.0590.
[17] R. Shokripour, J. Anvik, Z. M. Kasirun and S. Zamani, "A time-based approach to automatic bug report assignment", Journal of Systems and Software, vol. 102, **(2015)**, pp. 109-122, https://doi.org/10.1016/j.jss.2014.12.049.
[18] Y. Tian, D. Lo, X. Xia and C. Sun, "Automated prediction of bug report priority using multi-factor analysis", Empirical Software Engineering, vol. 20, no. 5, **(2014)**, pp. 1354-1383, https://doi.org/10.1007/s10664-014-9331-y.
[19] J. Uddin, R. Ghazali, M. M. Deris, R. Naseem and H. Shah, "A survey on bug prioritization", Artificial Intelligence Review, **(2016)**, pp. 1-36, https://doi.org/10.1007/s10462-016-9478-6.

[20] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen and X. Wang, "Improving Automated Bug Triaging with Specialized Topic Model", IEEE Transactions on Software Engineering, vol. 5589(c), **(2016)**, pp. 1-1, https://doi.org/10.1109/TSE.2016.2576454.
[21] X. Xia, D. Lo, E. Shihab and X. Wang, "Automated Bug Report Field Reassignment and Refinement Prediction", IEEE Transactions on Reliability, **(2015)**, pp. 1-20, https://doi.org/10.1109/TR.2015.2484074.
[22] J. Xuan, H. Jiang, Y. Hu, Z. Ren, W. Zou, Z. Luo and O. F. S. Oftware, "Supplemental Material Towards Effective Bug Triage with Software Data Reduction Techniques", vol. 27, no. 1, **(2015)**, pp. 1-6.
[23] J. Xuan, H. Jiang, Z. Ren, J. Yan and Z. Luo, "Automatic Bug Triage using Semi-Supervised Text Classification", Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), (60805024), **(2010)**.
[24] J. Zhang, X. Y. Wang, D. Hao, B. Xie, L. Zhang and H. Mei, "A survey on bug-report analysis", Science China Information Sciences, vol. 58, no. 2, **(2015)**, pp. 1-24, https://doi.org/10.1007/s11432-014-5241-2.
[25] T. Zhang, J. Chen, G. Yang, B. Lee and X. Luo, "Towards more accurate severity prediction and fixer recommendation of software bugs", Journal of Systems and Software, vol. 117, **(2016)**, pp. 166-184, https://doi.org/10.1016/j.jss.2016.02.034.
[26] W. Zhang, S. Wang and Q. Wang, "KSAP: An approach to bug report assignment using KNN search and heterogeneous proximity", Information and Software Technology, vol. 70, **(2016)**, pp. 68-84. https://doi.org/10.1016/j.infsof.2015.10.004.

