Sādhanā Vol. 42, No. 5, May 2017, pp. 655–669
DOI 10.1007/s12046-017-0629-5

Software bug prediction using object-oriented metrics

DHARMENDRA LAL GUPTA[1,2,*] and KAVITA SAXENA[2]

[1]Department of Computer Science and Engineering, Kamla Nehru Institute of Technology, Sultanpur 228118, India
[2]Department of Computer Science and Engineering, Mewar University, Chittorgarh 312901, India
e-mail: dlgupta2002@gmail.com; ksaxena72@yahoo.com

MS received 4 April 2016; revised 1 September 2016; accepted 3 November 2016

Abstract. Software quality is the fundamental requirement for a user, academia person, software developing organizations and researchers. In this paper a model for object-oriented Software Bug Prediction System (SBPS) has been developed. This model is capable of predicting the existence of bugs in a class if found, during software validation using metrics. The designed model forecasts the occurrences of bugs in a class when any new system is tested on it. For this experiment some open source similar types of defect datasets (projects) have been collected from Promise Software Engineering Repository. Some of these datasets have been selected for prediction of bugs, of which a few are not involved in model construction. First of all, we have formulated some hypotheses corresponding to each and every metric, and from metrics validation based on hypothesis basis finally 14 best suitable metrics have been selected for model creation. The Logistic Regression Classifier provides good accuracy among all classifiers. The proposed model is trained and tested on each of the validated dataset, including validated Combined Dataset separately too. The performance measure (accuracy) is computed in each case and finally it is found that the model provides overall averaged accuracy of 76.27%.

Keywords. Software bug; metrics; correlation of metrics and bug; software bug prediction.

1. Introduction

Software quality is the paramount need for a user, academia person, software developing organizations and researchers. "Quality is never an accident. It is the result of an intelligent effort" [1]. Prediction of software defect can only be possible either on the basis of historical data accumulated during implementation of similar or same software projects or it can be developed using design metrics collected during design phase of software development.

Object-oriented (OO) approach is different from the traditional programming approach. It separates data and control; it is based on objects, each of which is a set of defined data and a set of fixed predefined operations, which are known as methods that can be performed on data. The object-oriented (OO) approach has become a more important cornerstone of software engineering than structural design and functional decomposition. In last three decades the object-oriented technology has been widely accepted and object-oriented development is now so pervasive that there is no longer a question of its viability.

OO paradigm provides a new potential and better way to analyse a problem, design a solution and implement it in software engineering. The OO approach provides better reusability, reliability and maintainability than the traditional approach, which is based on functional decomposition. Encapsulation is the attribute of OO System that creates self-contained objects that are very easily incorporated into a new design, which will basically promote reusability [2].

Software metric has been defined by Paul Goodman as "The continuous application of measurement based techniques to the software development process and all its product[s] to provide meaningful and timely information, together with the use of those procedures to recover that process and its products". Mostly it is categorized into three broader categories: process metric, product metric and project metric. Process metric is used to improve software development and low product maintainability, e.g., defect removal effectiveness during development, the response time of the fix process and pattern of testing of defect arrival. Product metrics deal with the characteristics of a product like size, complexity, design features, presence of quality level in the product and performance of the product, whereas project metrics cover the number of staff members involved in software development, their staffing pattern throughout the life cycle of the software, cost incurred in development of the project, schedule managed and productivity gained [3].

*For correspondence

1.1 *Organization of the paper*

The paper is organized as follows. In Section 2 we provide background and related work on software quality prediction using software metrics and various software quality prediction models available in literature. Section 3 presents the methodology opted in this paper, in which the relationships between software metrics and the respective defects in terms of bugs are shown. All the open source datasets have been aggregated in one dataset named combined_dataset. The combined_dataset is then passed through validation test in SPSS tool [4]; on this validated combined dataset, Pearson correlation analysis is applied and the relationship among the various metrics and the respective bug value is analysed; 14 most influential metrics have been validated also through the hypothesis validation technique using p-values as well as values computed as coefficient of determination R^2. Subsequently 14 different classifiers (methodologies) named Naïve Bayes, LivSVM (Support Vector Machine), Logistic Regression, Multi-Layer Perceptron, SGD (Stochastic Gradient Descent), SMO (Sequential Minimal Optimization), Voted Perceptron, Attribute Selected Classifier, Logit Boost, Tree Decision Stamp, Random forest, Classification Via Regression, Random Tree and REP (Reduce Error Proning) Tree have been used; a 10-cross validation on validated combined dataset with 3597 instances has been applied and the accuracy of each classifier is computed; finally it is found that the Logistic Regression Classifier provides better accuracy among all these classifiers.

Section 4 reveals the pre-processing and model construction. In section 5, 13 different experimental works have been done in which the cross project training–testing phenomena has been applied using the proposed model based on Logistic Regression Classifier.

Section 6 depicts conclusions of all the various experiments performed earlier. Limitations and future work are mentioned in section 7.

2. Background and related work

Software quality has been under study for a very long time. Various software quality models are available in the literature. In the last few decades many software developers and researchers studies have been carried out in this area of software quality prediction. Neural networks, fuzzy logic, regression tree, etc. and their applications have been used for software quality prediction.

Chidamber and Kemerer [5] presented a theoretical work focusing on OO metrics at design level of Software Development Life Cycle. These metrics are based upon the measurement theories and are used by well-experienced OO software developers.

Chidamber and Kemerer [6] focused on the key requirements of measurement to improve the quality of software with the help of a new metrics suite that consists of six design level metrics named WMC, DIT, NOC, CBO, RFC and LCOM.

Emam *et al* [7] used the Chidamber and Kemerer [5] metrics suite and a subset of Lorenz and Kidd metrics to compute the total impact of class size on validation of OO metrics. The prime consent of the author in this paper was on the prediction of class fault proneness for the faults found in the field. The authors have provided strong evidence showing confounding effect of size in the product metrics to fault-proneness relationship. They have experimentally examined and justified an empirical methodology for examining whether there is a confounding effect. Finally, they have performed a study using a large C++ system to justify the confounding effect that arises.

Gursaran and Roy [8] noticed that Weyuker's property 9 is not satisfied by any inheritance metric in the Chidamber and Kemerer [6] metric suite. The authors evaluated inheritance metrics proposed by Brito and Carapuca [9]. The authors showed by building on the metric assumptions and the definitions of concatenation given by Chidamber and Kemerer [6] that a particular class of structural inheritance metrics defined on a diagraph abstraction of the inheritance structure can never satisfy property 9 of Weyuker.

Jureczko and Madeyski [10] have presented a review based on process methods, i.e., number of revisions, number of changed lines that are new and number of defects in previous revision. They analysed and said that all of the aforementioned metrics rely on the information corresponding to source code artifact changes. It is also pointed out that some of the changes may not be directly related to the software process used. Finally the authors have shown that process metrics can be an effective addition to software defect prediction modules usually built upon prediction metrics.

Jin *et al* [11] proposed fuzzy c-means clustering (FCM) and radial basis function neural network (RBFNN) to build a prediction model of the fault proneness; RBFNN is used as a classificatory, and FCM as a cluster.

Jureczko and Madeyski [12] have presented an analysis regarding defect prediction using clustering technique on software projects. They have used a data repository with 92 versions of 38 proprietary, academic and open source projects. In this paper Hierarchical and K-mean clustering, as well as Kohonen's neural network, has been used to find the groups of similar projects. Two defect prediction models were created for each of the identified groups. In this study Ckjm tool has been used for the retrieval of all metrics, which will be used for the defect prediction model. JUnit and FitnNess have been used as test tools. The authors have identified two clusters and compared results obtained by other researchers. Finally clustering is suggested and applied.

Catal [13] reported in his paper about a complete survey of 90 research papers regarding the direction of research

work done during 1990–2009. In addition to it he has also focused on machine learning approach and statistical data calculation for the fault prediction.

Okutan and Yildz [14] applied Bayesian network to determine the probabilities of occurrences of faults and relation with metrics. The authors used the Promise data repository; they defined two new metrics— NOD (Number of developers) and Lack of coding quality (LOCQ)—for the source code quality. On the basis of their experiment, they noticed that RFC, LOC and LOCQ are the most effective metrics whereas CBO, WMC and LCOM metrics are very less effective, and even metrics NOC and DIT are not much effective.

3. Methodology

3.1 *Data description*

In this paper, 12 open source similar types of projects (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci) have been selected, which are the latest release of their retrospective versions and are collected from Promise Software Engineering Repository, which is available at https://code.google.com/p/promisedata/w/list [15] and shown in table 1. The datasets considered for the experimental work in this thesis are of OO defect projects and Marian Jureczko datasets (https://code.google.com/p/promisedata/wiki/MarianJureczko) [16].

3.2 *Data assumption*

The proposed model will consider and work efficiently only on OO datasets. It is presumed that all the factors that may affect the system in buggy and non-buggy categories will be presented in workable dataset/datasets.

All the defect datasets must be related with their different metrics sets and bug values corresponding to each and every instance of the dataset/datasets. If any instance lacks the value of some metrics then in that case all such metrics values will be considered as zero.

Table 1. Data description.

Sl. no.	Dataset name	Instances
1	Camel1.6	965
2	Tomcat 6.0	858
3	Ant 1.7	745
4	jEdit4.3	492
5	Ivy 2.0	352
6	arc	234
7	e-learning	64
8	berek	43
9	forrest 0.8	32
10	Zuzel	29
11	Intercafe	27
12	Nieruchomosci	27

3.3 *Data validation*

The different defect datasets collected from Promise repository having duplicate data should be deleted except for one entry. Only then will any module provide unbiased result, and the actual performance of the module may be computed.

To avoid duplicate entries, all the datasets mentioned earlier have been validated. The data validation has been performed using the SPSS tool [4] separately on each dataset taken under study.

The same technique has also been applied on Combined_dataset, which is an aggregation of datasets (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci) with 3868 instances, and all other datasets used in this thesis separately too to avoid duplicate entries. After data validation applied on the Combined_datset, total unique instances are only 3597 (of validated Combined_-dataset) and remaining 271 duplicate entries have been discarded.

3.4 *Metrics description*

3.4a *Dependent and independent variables*: Here, bug is a dependent variable that shows whether there is any bug in a class of the each dataset or not. The independent variables are WMC, DIT, NOC, CBO, RFC and LCOM [6], CA and CE [1], LOC and LCOM3 [17] and NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC and AVG-CC [12].

3.5 *Metrics assumptions*

There are some assumptions regarding metrics for consideration during construction of software bug prediction model to get better response, which are mentioned below.

i. Input to the model will be from design as well as coding level metrics of OO software system.
ii. The model is based on 20 independent OO metrics considered under study, which are WMC, DIT, NOC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC and AVG_CC.
iii. It is assumed that if any dataset fails to have all above-mentioned metrics then that metric value may be assumed to be zero.
iv. The dependent variable (bug value) related to each instance takes two values, either 0 (zero) or 1(one);

here zero bug value is assumed for non-buggy and one bug value is considered for buggy instance. In this study it is assumed that those instances whose bug values are zero are zero and every none-zero bug value instance has been assumed as one.

v. The proposed model's classifier (Logistic Regression) will only consider the bug value as binary: either buggy or non-buggy.

vi. The training and testing dataset must be compatible to the proposed model. This means both the datasets considered under training and testing phases must have the same metric suite and number of metrics.

vii. All the metric validations under this paper have been done on the significance level $\alpha = 0.05$, p-values < 0.0001 and $R^2 \geq 0.005$.

viii. To predict the occurrences of bug on proposed model using some new OO softwares of the real world it is assumed that the bug values related to each instance are either zero or one.

3.6 *Metrics validation*

Metrics used in the model are validated and selected on the basis of Pearson correlation. The step by step key process for metric validation and selection of the most suitable metrics that has been used under study is discussed below.

First of all we collected 12 datasets (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci), combined all these datasets, called it Combined Dataset, with total 3868 instances, and applied data validation using the SPSS tool [4]. The validated Combined Dataset has only 3597 instances (details are mentioned in the Data validation section).

3.7 *Hypotheses formulation*

Twenty Null hypotheses have been formulated, namely Hypo_wmc, Hypo_dit, Hypo_noc, Hypo_cbo, Hypo_rfc, Hypo_lcom, Hypo_ca, Hypo_ce, Hypo_npm, Hypo_lcom3, Hypo_loc, Hypo_dam, Hypo_moa, Hypo_mfa, Hypo_cam, Hypo_ic, Hypo_cbm, Hypo_amc, Hypo_max_cc and Hypo_avg_cc. It is presumed that each metric cannot predict the fault proneness of a class.

3.8 *Pearson correlation on validated Combined Dataset*

Pearson correlation is applied at significance level $\alpha = 0.05$ on validated Combined Dataset and the most important and relevant relation is found between each metric and its respective bug value separately.

From table 2 it is clear that metrics WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, AMC, MAX_CC and AVG_CC show bug values 0.223, 0.157, 0.297, 0.081, 0.090, 0.203, 0.181, –0.073, 0.232, 0.125, 0.193, −0.200, 0.052, 0.191 and 0.116, respectively, which are in bold and are different from 0 (zero) with a significance level $\alpha = 0.05$. Therefore these metrics values emerged to be significantly correlated to the number of bugs within a class (instance). To consider the most prominent metrics in this metrics set, AMC metric has been discarded because it has value 0.052, which is very close to significance level α (0.05) and the remaining metrics have bug values greater than significance level α considered. Therefore the final metrics set that is collected from this table is a collection of metrics WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC with their bug values 0.223, 0.157, 0.297, 0.081, 0.090, 0.203, 0.181, –0.073, 0.232, 0.125, 0.193, −0.200, 0.191 and 0.116, respectively.

It is seen from table 2 that metrics LCOM3 and CAM show negative values, which means they have negative impact on the fault proneness. This analysis exhibits that fault proneness increases with a decreases in cohesion.

It is also noticeable that the metrics DIT, NOC, MFA, IC, CBM and AMC have bug values that are not much greater than significance level α; therefore they were excluded from further analysis in this paper.

Hence, from table 3, hypotheses are validated in terms of p-values and finally 14 metrics WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM,

Table 2. Metrics and the correlated bug of validated Combined Dataset.

Sl. no.	Variables	Bug count
1	WMC	**0.223**
2	DIT	0.043
3	NOC	0.037
4	CBO	**0.157**
5	RFC	**0.297**
6	LCOM	**0.081**
7	CA	**0.090**
8	CE	**0.203**
9	NPM	**0.181**
10	LCOM3	**–0.073**
11	LOC	**0.232**
12	DAM	**0.125**
13	MOA	**0.193**
14	MFA	–0.006
15	CAM	**–0.200**
16	IC	0.038
17	CBM	0.025
18	AMC	0.052
19	MAX_CC	**0.191**
20	AVG_CC	**0.116**
21	Bug count	**1**

Most influential values for bug prediction are in bold.

MAX_CC and AVG_CC selected for model designing, which will be able to predict the most-fault-prone classes in the datasets.

In terms of coefficient of determination (R^2), these metrics (WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC) also show a strong correlation with bug value (table 3).

3.9 *Classifier selection*

The stepwise classifier selection process is as follows.

i. Twelve different OO defect datasets (Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci) have been selected from Promise open source repository.
ii. Combine all the datasets mentioned in step i one by one in an aggregation manner.
iii. Data validation process has been applied on Combined Dataset using the SPSS tool [4]. In this validation process, datasets will be free from ambiguity. Duplicate entries will be deleted and only one representative of each instance value is considered.
iv. Pearson correlation technique is applied on validated Combined Dataset. In this process, p-values and R^2 (coefficient of determination) values are computed at significance level $\alpha = 0.05$. On the basis of p-values and R^2 values the 20 metrics are validated and only 14 most prominent metrics (WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC) whose p-values $<$ 0.0001 have been considered for the study.
v. Select values of only 14 metrics (mentioned in step iv) and related bug values from validated Combined Dataset.
vi. Apply all 14 different classifiers Naïve Bayes, LivSVM (Support Vector Machine), Logistic Regression, Multi-Layer Perceptron, SGD (Stochastic Gradient Descent), SMO (Sequential Minimal Optimization), Voted Perceptron, Attribute Selected Classifier, Classification Via Regression, Logit Boost, Tree Decision Stamp, Random forest, Random Tree and REP (Reduce Error Pruning) Tree on validated Combined Dataset with 14 most prominent metrics at 10-cross validation.
vii. Step vi will produce Logistic Regression as the most prominent classifier with maximum accuracy 85.04% (table 4).

Table 3. Relationship between object-oriented metrics and bug values based on Pearson correlation analysis on validated Combined Dataset.

		Bug values	
Sl. no.	Variables	1. p-values	2. R^2
1	WMC	**0.000**	**0.050**
2	DIT	0.009	0.002
3	NOC	0.025	0.001
4	CBO	**<0.0001**	**0.025**
5	RFC	**<0.0001**	**0.088**
6	LCOM	**<0.0001**	**0.007**
7	CA	**<0.0001**	**0.008**
8	CE	**<0.0001**	**0.041**
9	NPM	**<0.0001**	**0.033**
10	LCOM3	**<0.0001**	**0.005**
11	LOC	**<0.0001**	**0.054**
12	DAM	**<0.0001**	**0.016**
13	MOA	**<0.0001**	**0.037**
14	MFA	0.728	0.000
15	CAM	**<0.0001**	**0.040**
16	IC	0.021	0.001
17	CBM	0.130	0.001
18	AMC	0.002	0.003
19	MAX_CC	**<0.0001**	**0.036**
20	AVG_CC	**<0.0001**	**0.013**
21	Bug	**0**	**1**

Most influential values for bug prediction are in bold.

4. Pre-processing

Terminology of raw data collection made suitable (normalized and pre-processed) for application in our designed model is discussed.

Step I — First of all select all open source raw datasets (OO defect datasets) named Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe and Nieruchomosci with instance vales 965, 858, 745, 492, 352, 234, 64, 43, 32, 29, 27 and 27, respectively, selected for the study in which all are in the decreasing order of their instance values. Apart from these datasets there are also some other similar datasets, named Wspomanagie, pbeans, Velocity, System data Management and Termo Project, considered in this thesis for the retrospective purpose as and when required

Step II — Select only 14 metrics WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC, which are the most prominent metrics for the bug prediction in all the datasets mentioned in step I

Step III — Validate only those datasets on which the proposed model is going to be tested

Step IV — Since the raw datasets are in .CSV (Comma separated value) format it has been changed into .arff (attribute relation file format), which is acceptable to WEKA, the data mining software

Table 4. Classifier selection values.

Sl. no.	Classifiers	Accuracy at 10-cross validation on validated Combined Dataset
1	Naïve Bayes	82.46
2	Lib SVM	84.65
3	Logistic Regression	85.04
4	Multi-Layer Perceptron	84.49
5	SGD	84.68
6	SMO	84.68
7	Voted Perceptron	84.40
8	Attribute Selected Classifier	84.63
9	Classification Via Regression	84.88
10	Logit Boost	85.02
11	Tree Decision Stamp	84.68
12	Random forest	84.57
13	Random Tree	78.26
14	REP Tree	84.04

Step V Logistic Regression Classification techniques are applied on all the validated datasets on only 14 selected metrics set, which are mentioned in step-II

Step VI Initially set the bug value field to either zero or one to make metrics compatible during the testing phase under prediction process

If (p-value < 0.0001)
{
Select the M_i
If (p-value$\rightarrow M_i$ == $R^2 \rightarrow M_i$) $1 \le i \le m$, $0 \le R^2 \le 1$
{
Return M_i
}
else
{
Reject M_i
}
else
{
Reject M_i
}
}

4.1 *Algorithm of the proposed model*

The algorithm of the proposed software bug prediction model is as follows.

Input: A set of datasets (D_1,D_2,D_3,…,D_n); D_v is a validated Combined Dataset; a set of metrics ($M = M_1, M_2, M_3,\ldots,M_m$); a set of classifiers (C_1,C_2,C_3,…,C_k).

Output: Overall averaged accuracy of the proposed model.

Algorithm steps

1. Select different OO defect datasets (D_1, D_2, D_3,…, D_n)
2. Combine all the datasets one by one
   $D = \sum_{i=1}^{n} D_i$
3. Remove duplicate entries using SPSS
   $D_v \leftarrow$ Apply data validation (D)
4. p-values, $R^2 \leftarrow$ Apply Pearson corelation (D_v)
5. Check the *p-values* and R^2 (coefficient of determination) values of each independent metric ($M = M_1, M_2, M_3,\ldots, M_m$) to dependent (bug) metric and follow these steps:
6. $M_s \leftarrow$ Select the set of metrics from step 5
7. Apply different classifiers C = (C_1, C_2, C_3,…,C_k) using M_s on D_v at 10-cross validation
8. $A_i \leftarrow$ Calculate accuracy of C_i on D_v $\forall i \in \{1, 2, 3, \ldots, k\}$
9. $A_{max} \leftarrow Max(A_i)$
10. $C_f \leftarrow$Select most prominent C_j ($1 \le j \le k$) where $A_j = A_{max}$
11. Remove duplicate entries from each dataset D_i using SPSS
    $D_i \leftarrow$ Apply data validation$(D_i) \forall i \in \{1, 2, \ldots, n\}$
12. Apply C_f on each dataset obtained from step11 using M_s for training and testing purpose for proposed model
13. Train the model on each validated dataset (D_1,D_2,D_3,…, D_n) obtained from step 11 and test on all datasets, including validated Combined Dataset, separately
14. Calculate the average accuracy of the model

4.2 *Assumptions of the proposed model*

i. This model will work on OO software (dataset) more effectively.
ii. This model has been designed on WEKA 3.7.11; therefore it will provide better result on this version of WEKA—a machine learning tool.
iii. Data must be pre-processed if the data format is not in .arff; it is required to first of all convert the new

dataset into .arff format, which is acceptable to it (pre-processing is mentioned in an earlier section).

iv. This model will produce better result on the set of 20 metrics (WMC, DIT, NOC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC and AVG-CC) considered in this study.

v. It is assumed that there is an actual relationship between bug (dependent variable) and OO metrics (independent variable).

vi. It is assumed that there is a clear dependency relation, which means that always a dependent variable is a function of an independent variable.

vii. It is assumed that the analysis is used to predict bug values in the range of validated Combined Dataset and not outside it, for which it is valid.

4.3 *Model construction*

From the hypotheses validation it is found that the bug will be highly correlated to the metrics WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC. The bug prediction model equation is presented later, where bug is the dependent variable whereas WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, CAM, MAX_CC and AVG_CC are independent and these are the most fruitful parameters to detect the bug of a class. It provides the probability of occurrence of bugs in the dataset that is to be tested.

The architecture of Data Validation, Metric selection and Classifier selection is shown in figure 1.

Figures 2 and 3 present block diagram of Data Pre-processing and Accuracy calculation using the proposed model.

$$\begin{aligned}\text{Pred(Bug)} = 1/(1 + \exp((&2.70517 + 0.01834 * \text{wmc} \\ &+ 0.01407 * \text{cbo} + 0.01309 * \text{rfc} - 0.00026 \\ &* \text{lcom} - 0.00951 * \text{ca} + 0.00127 * \text{ce} \\ &- 0.01822 * \text{npm} + 0.27841 * \text{lcom 3} \\ &- 0.00021 * \text{loc} + 0.57059 * \text{dam} + 0.02130 \\ &* \text{moa} - 0.76284 * \text{cam} + 0.02210 * \text{max\_cc} \\ &- 0.05913 * \text{avg\_cc})))\end{aligned} \tag{1}$$

4.4 *Confusion matrix*

Confusion matrix is basically used to measure the performance of two classes (correctly classified and incorrectly classified) problem for a given dataset, as it is shown in table 5, where the right diagonal elements TP and TN are correctly classified instances as well as FP and FN are incorrectly classified

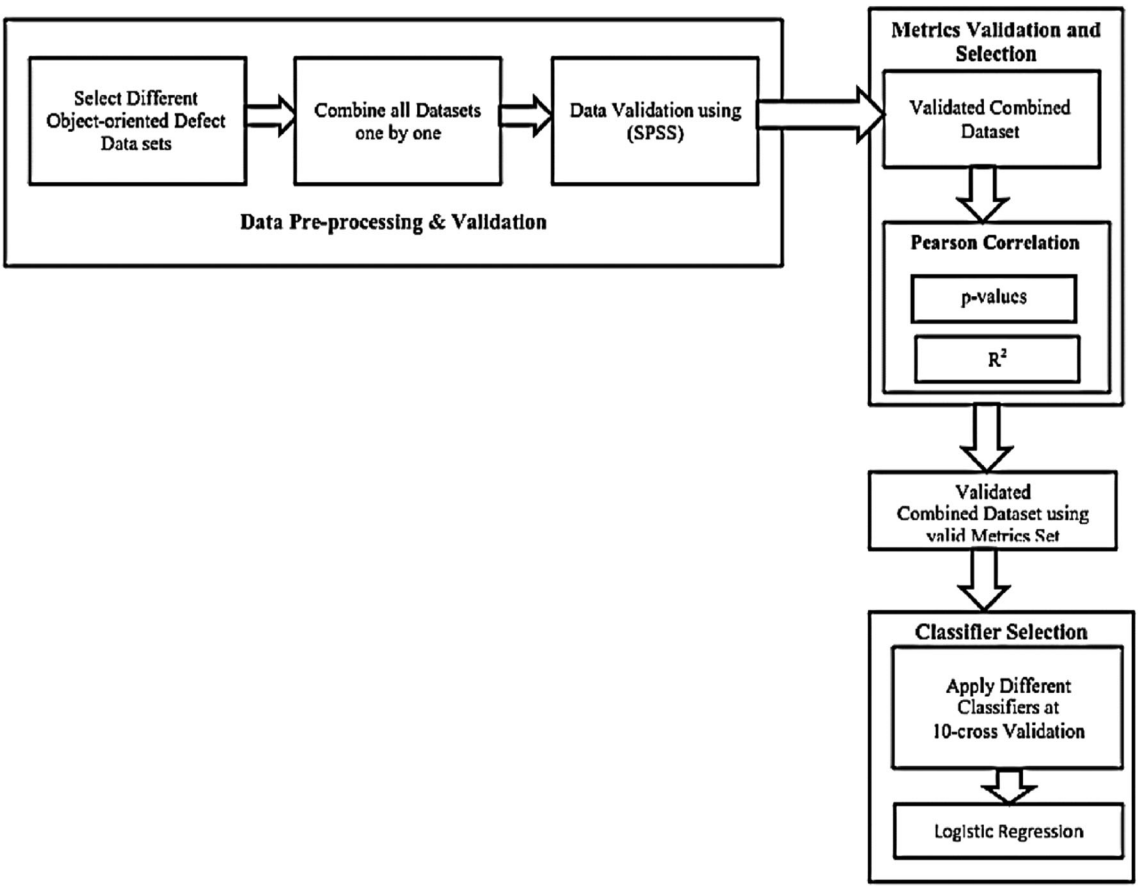

Figure 1. Architecture of data validation, metrics selection and classifier selection.

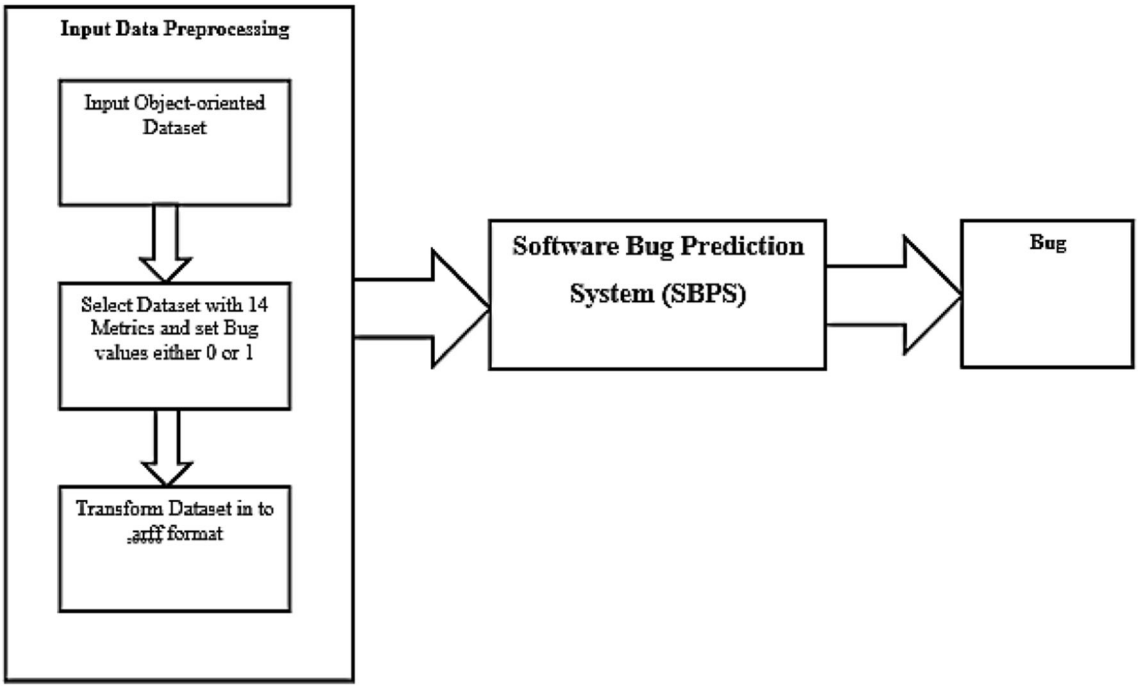

Figure 2. Block diagram of proposed model.

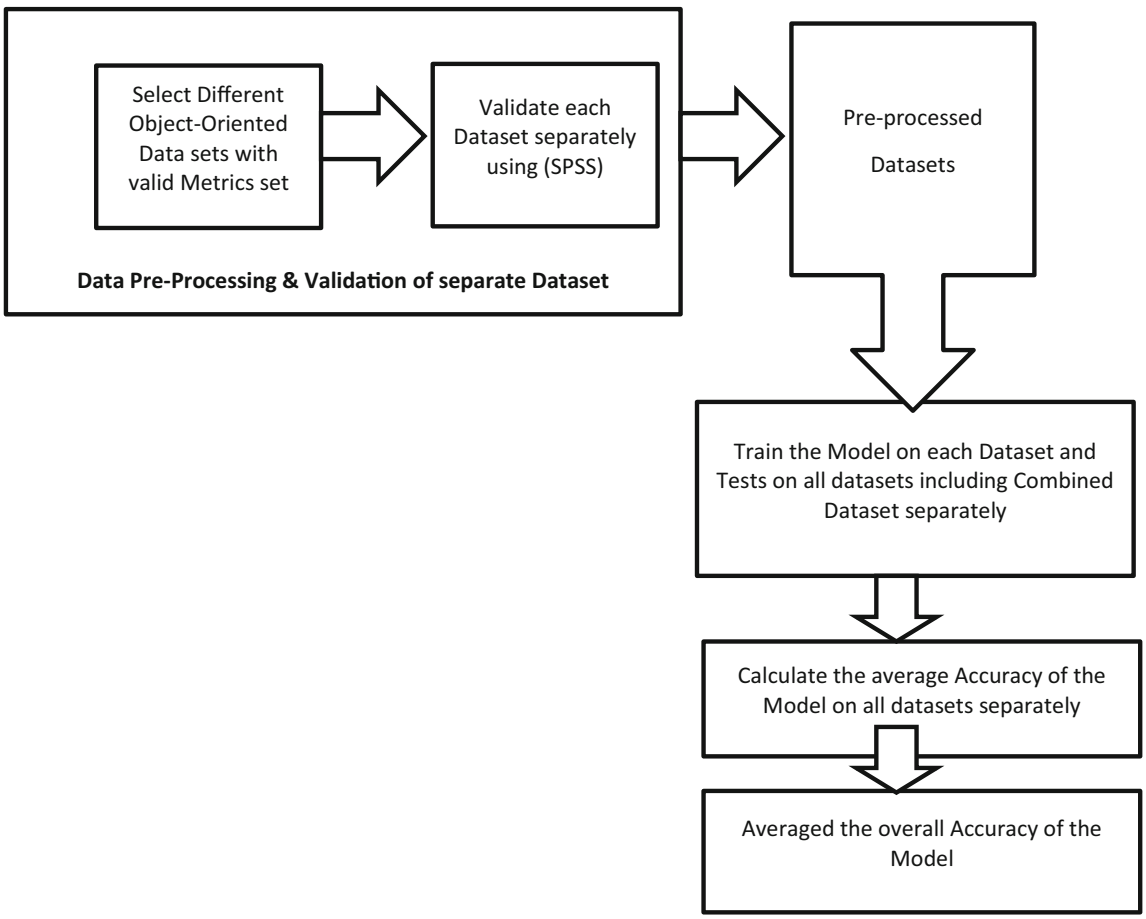

Figure 3. Block diagram of Data Pre-processing and Accuracy calculation of the proposed model.

Table 5. Confusion matrix.

	Predicted	
Actual	Non-buggy	Buggy
Non-buggy	TP	FN
Buggy	FP	TN

TP is True Positive (correctly classified as non-buggy, which are actually non-buggy instances)

TN is True Negative (correctly classified as buggy, which are actually buggy instances)

FP is False Positive or False Alarm or Type-I error (α) (actually buggy instances have been classified as non-buggy)

FN is False Negative or Type-II error (β) (actually non-buggy predicted as buggy)

Total number of instances = correctly classified instance + incorrectly classified instance

Correctly classified instance = TP + TN

Incorrectly classified instance = FP + FN

instances. It is a table of rows and columns in which rows represent the actual classes and columns represent the predicted classes. It is also called as coincidence matrix. This matrix has been used here for performance evaluation of designed model using Logistic Regression Classifier applied on all the datasets. The performance evaluation technique has been taken from Witten and Frank [18] (https://code.google.com/p/promisedata/wiki/MarianJureczko [16]) and shown in terms of accuracy [13], which is computed as follows:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{correctly classified instances}}{\text{total number of instances}}. \quad (2)$$

5. Experimental work

5.1 *Experiment No. 1*

The model is trained on validated dataset Camel 1.6 with 884 instances and 79.98% accuracy and tested an all the datasets Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe, Nieruchomosci and Combined Dataset. Table 8 depicts the following results.

- In this experiment when the proposed model is trained on dataset Camel 1.6 and tested on all the 13 datasets, the model provides the accuracy 79.98%, 74%, 78.45%, 84.24%, 80.87%, 86.05%, 91.23%, 76.74%, 90.23%, 58.62%, 81.48%, 69.23% and 79.04%, correspondingly, during the testing phase.
- It is also found that dataset Zuzel provides 58.62% accuracy, which is the minimum one, when treated as test set while training of the model was done on Camel 1.6.
- Maximum accuracy of 91.23% is found on dataset e-learning during testing, when the model is trained on the dataset Camel 1.6.
- Datasets: e-learning (91.23), forrest 0.8 (90.32), arc (86.05), jEdit4.3 (84.24), Intercafe (81.48), Ivy 2.0 (80.87), Camel 1.6 (79.98), Combined Dataset (79.04), Ant 1.7 (78.45), berek (76.74), Tomcat 6.0 (74), Nieruchomosci (69.23) and Zuzel (58.62). e The accuracy is provided in per-centage in decreasing order for the test sets when training is done on dataset Camel 1.6.
- Finally this experiment provides 79.25% average accuracy during prediction (testing) when training is done on Camel 1.6.

5.2 *Experiment No. 2*

In this experiment the model is trained on Tomcat 6.0 with 796 instances and testing is applied on all the datasets mentioned. Predicted accuracies are presented in tables 6 and 7. From table 8 the following results are tabulated.

- The 13 datasets (datasets Camel1.6, Tomcat 6.0, Ant 1.7, jEdit4.3, Ivy 2.0, arc, e-learning, berek, forrest 0.8, Zuzel, Intercafe, Nieruchomosci and Combined Dataset set) provide model accuracy of 79.64%, 90.83%, 78.59%, 92.23%, 88.99%, 88.37%, 91.23%, 72.09%, 93.55%, 65.52%, 81.48%, 61.54% and 84.65%, respectively, during testing.
- Maximum accuracy of 93.55% is found during testing of the model when training was done on the dataset Tomcat 6.0.
- It is also seen that dataset Nieruchomosci provided minimum 61.54% accuracy during testing.
- The sequence showing decreasing order of accuracy when testing is done on all the datasets is forrest 0.8, jEdit 4.3, e-learning, Tomcat 6.0, Ivy 2.0, arc, Combined Dataset, Intercafe, Camel 1.6, Ant 1.7, berek,

Table 6. Predicted accuracy of dataset Tomcat 6.0 (assuming that all bug values are zero).

Trained on validated Combined Dataset with accuracy (%)		Predicted (tested) on dataset Tomcat 6.0 with accuracy (%) (assuming all bug values are one)	
Instances	Accuracy (%)	Instances	Accuracy (%)
3597	85.18	829 (non-buggy) 29 (buggy)	96.62 3.38

Table 7. Predicted accuracy of dataset Tomcat 6.0 (assuming that all bug values are one).

Trained on validated Combined Dataset with accuracy (%)		Predicted (tested) on dataset Tomcat 6.0 with accuracy (%) (assuming that all bug values are zero)	
Instances	Accuracy (%)	Instances	Accuracy (%)
3597	85.18	829 (non-buggy) 29 (buggy)	96.62 3.38

Zuzel and Nieruchomosci with their accuracy values of 93.55%, 92.23%, 91.23%, 90.83%, 88.99%, 88.37%, 84.65%, 81.48%, 79.64%, 78.59%, 72.09%, 65.52% and 61.54%, respectively.
- Finally the model provided 82.21% average accuracy in this experiment.

5.3 *Experiment No. 3*

In this experiment the model is trained on Ant 1.7 with 724 instances and testing is done on all the datasets, including Combined Dataset too. Following points are made from the assessment of this experiment presented in table 8.

- During the testing process the above-mentioned 13 datasets (mentioned in section 4.1) provide model accuracy of 78.28%, 85.18%, 82.6%, 83.61%, 85.51%, 88.73%, 91.23%, 88.37%, 90.32%, 58.62%, 77.78%, 61.54% and 82.49%, respectively.
- It is also discovered that dataset e-learning provides maximum accuracy of 91.23% whereas only 58.62% accuracy is manifested by dataset Zuzel, which is the minimum value.
- Datasets e-learning, forrest 0.8, arc, berek, Ivy 2.0, Tomcat 6.0, jEdit4.3, Ant 1.7, Combined Dataset, Camel 1.6, Intercafe, Nieruchomosci and Zuzel provide 91.23%, 90.32%, 88.73%, 88.37%, 85.51%, 85.18%, 83.61%, 82.6%, 82.49%, 78.28%, 77.78%, 61.54% and 58.62% accuracy value, respectively, in decreasing order during testing.
- Finally the model provides 81.10% accuracy value in this experiment.

5.4 *Experiment No. 4*

The model is trained on dataset jEdit4.3 with 476 instances and 98.74% accuracy and tested on all the datasets mentioned in earlier section.

- Table 8 depicts that the values of accuracy when applied on all the datasets under prediction varies, and no specific pattern is shown by the model.
- It is noticeable that dataset jEdit4.3 provides maximum accuracy of 98.74% during testing when training was done on the same dataset.
- It is also seen that dataset Zuzel prompts minimum accuracy in testing mode of 55.17% when training is done on jEdit4.3.
- Datasets jEdit 4.3, forrest 0.8, e-learning, Tomcat 6.0, Intercafe, Ivy 2.0, arc, Combined Dataset, Camel 1.6, Ant 1.7, berek, Nieruchomosci and Zuzel show prediction accuracy of the model in decreasing order of 98.74%, 93.55%, 91.23%, 89.45%, 88.89%, 88.41%, 87.44%, 84.65%, 79.19%, 77.49%, 65.12%, 61.54% and 55.17%, respectively.
- On average, dataset jEdit4.3 shows 81.61% accuracy.

5.5 *Experiment No. 5*

In this experiment the model is now trained on Ivy 2.0 with 345 instances and 90.43% accuracy and testing is done on all the datasets separately. Table 8 reveals the computed results, which are listed here.

- Dataset Camel 1.6 to Combined Dataset mentioned in section 4.1 provide accuracy at testing phase of 79.41%, 89.57%, 80.66%, 93.07%, 90.43%, 89.3%, 91.23%, 79.07%, 93.55%, 68.97%, 81.48%, 61.54% and 85.15%.
- It is seen from table 8 that the datasets forrest 0.8, jEdit4.3, e-learning Ivy 2.0, Tomcat 6.0, arc, Combined Dataset, Intercafe, Ant 1.7, Camel 1.6, berek, Zuzel and Nieruchomosci provide accuracy of 93.55%, 93.07%, 91.23%, 90.43%, 89.57%, 89.3%, 85.15%, 81.48%, 80.66%, 79.41%, 79.07%, 68.97% and 61.54%, respectively, in decreasing order of their values.
- Dataset Nieruchomosci provides minimum prediction accuracy of 61.54%, whereas dataset forrest 0.8 provides maximum accuracy of 93.55%, when testing is done on it.
- In this case the average accuracy of the model is 83.34%.

5.6 *Experiment No. 6*

In this case of experimental work, the model is trained on dataset arc with 215 instances and 90.23% accuracy, whereas testing is done on all the datasets mentioned in the earlier section, which can be seen in table 8.

Table 8. Accuracy measured on training and testing datasets using Logistic Regression Classifier.

Training and testing both on normalized datasets

Sl. no.	Training on datasets with accuracy (%) Datasets	Instances	Accuracy (%)	Testing on datasets with accuracy (%) Camel 1.6 884	Tomcat 6.0 796	Ant 1.7 724	jEdit 4.3 476	Ivy 2.0 345	Arc 215	e-learning 57
1	Camel 1.6	884	79.98	79.98	74	78.45	84.24	80.87	86.05	91.23
2	Tomcat 6.0	796	90.83	79.64	90.83	78.59	92.23	88.99	88.37	91.23
3	Ant 1.7	724	82.6	78.28	85.18	82.6	83.61	85.51	88.73	91.23
4	jEdit 4.3	476	98.74	79.19	89.45	77.49	98.74	88.41	87.44	91.23
5	Ivy 2.0	345	90.43	79.41	89.57	80.66	93.07	90.43	89.3	91.23
6	Arc	215	90.23	78.51	37.57	77.76	96.85	76.52	90.23	91.23
7	e-learning	57	100	61.99	69.22	57.73	63.24	66.38	66.51	100
8	berek	43	100	77.26	82.29	80.94	78.57	86.67	87.44	80.7
9	forrest 0.8	31	100	65.05	76.76	75.14	82.35	81.74	76.74	85.96
10	zuzel	29	96.55	71.72	60.93	70.03	66.81	72.46	72.09	78.95
11	Intercafe	27	100	49.77	75.63	55.25	52.52	62.03	73.49	64.91
12	Nieruchomosci	26	100	72.96	57.54	68.65	43.07	59.71	80	78.95
13	Combined Dataset	3597	85.18	79.3	90.7	79.42	96	89.28	87.91	91.23

Average = 76.27

Training and testing both on normalized datasets

Sl. no.	Testing on datasets with accuracy (%) berek 43	forrest 0.8 31	Zuzel 29	Intercafe 27	Nieruchomosci 26	Combined Dataset 3597	Min.	Max.	Average
1	76.74	90.32	58.62	81.48	69.23	79.04	58.62	91.23	79.25
2	72.09	93.55	65.52	81.48	61.54	84.65	61.54	93.55	82.21
3	88.37	90.32	58.62	77.78	61.54	82.49	58.62	91.23	81.1
4	65.12	93.55	55.17	88.89	61.54	84.65	55.17	98.74	81.61
5	79.07	93.55	68.97	81.48	61.54	85.15	61.54	93.55	83.34
6	67.44	90.32	55.17	81.48	61.54	71.78	37.57	96.85	75.11
7	53.49	48.39	48.28	55.56	61.54	63.25	48.28	100	62.74
8	100	83.87	82.76	74.07	61.54	80.87	61.54	100	81.31
9	79.07	100	75.86	74.07	57.69	74.65	57.69	100	77.31
10	79.07	74.19	96.55	51.85	65.38	68.39	51.85	96.55	71.42
11	74.42	32.26	44.83	100	76.92	59.72	32.26	100	63.21
12	86.05	77.42	75.86	55.56	100	64	43.07	100	70.75
13	74.42	93.55	55.17	85.19	61.54	85.18	55.17	96	82.22

Average = 76.27

- The accuracy provided by the model during testing phase of all the 12 datasets and Combined Dataset too are 78.51%, 37.57%, 77.76%, 96.85%, 76.52%, 90.23%, 91.23%, 67.44%, 90.32%, 55.17%, 81.48%, 61.54% and 71.78%, respectively.
- The model accuracy in this experiment in decreasing order of datasets is as follows: jEdit4.3—96.85%, e-learning—91.23%, forrest 0.8—90.32%, arc—90.23%, Intercafe—81.48%, Camel1.6—78.51%, Ant 1.7—77.76%, ivy2.0—76.52%, Combined Datset—71.78, berek—67.44%, Nieruchomosci—61.54%, Zuzel—55.17% and Tomcat 6.0—37.57%.
- It is found that dataset Tomcat 6.0 provides minimum accuracy of 37.57%.
- The maximum accuracy value is achieved on jEdit4.3 of 96.85% during testing of the model.
- On average the model prompts 75.11% accuracy in this case.

5.7 *Experiment No. 7*

In this case the model is trained on dataset e-learning with 57 instances and 100% accuracy and testing is done on all the datasets, including e-learning also. From table 8, following points are made.

- Predicted accuracy values of the model in this empirical study are 61.99%, 69.22%, 57.73%, 63.24%, 66.38%, 66.51%, 100%, 53.49%, 48.39%, 48.28%, 55.56%, 61.54% and 63.25% of datasets Camel 1.6 to Combined Dataset in the sequence as mentioned in earlier sections.
- It is seen that the dataset Zuzel shows minimum accuracy of 48.28%.
- Dataset e-learning provides the maximum accuracy of 100% when testing is done.
- In decreasing order of accuracy values of datasets e-learning, Tomcat 6.0, Arc, Ivy 2.0, Combined Dataset, jEdit4.3, Camel 1.6, Nieruchomosci, Ant 1.7, Intercafe, berek, forrest 0.8 and Zuzel the values are 100%, 69.22%, 66.51%, 66.38%, 63.25%, 61.99%, 57.73%, 55.56%, 53.49%, 48.39% and 48.28%, respectively.
- However datasets jEdit4.3 and Combined Dataset provide 63.24% and 63.25% accuracy of the model at testing stage, which are likely to be equal values of accuracy.
- The average value of the accuracy in this case is 62.74%.

5.8 *Experiment No. 8*

The model is trained on dataset berek with 43 instances and 100% accuracy and tested on all the datasets mentioned in the previous section.

- From table 8 it is seen that the values of accuracy when applied on all the datasets under prediction vary, and no specific pattern is shown by the model.
- It is noticeable that dataset berek provides maximum accuracy of 100% during testing, when training is done on the same dataset.
- It is also seen that dataset Nieruchomosci prompts minimum accuracy during testing of 61.54% when training is done on dataset berek.
- Datasets berek, arc, Ivy 2.0, forrest 0.8, Zuzel, Tomcat 6.0, e-learning, jEdit4.3, Camel 1.6, Intercafe and Nieruchomosci provide accuracy of the model in decreasing order of 100%, 87.44%, 86.67%, 83.87%, 82.76%, 82.29%, 80.7%, 78.57%, 77.26%, 74.07% and 61.54%, respectively.
- Datasets Ant 1.7 and Combined Dataset provide quite similar model accuracies, which are 80.94% and 80.87 %, respectively.
- On average, Dataset-8 shows 81.31% accuracy.

5.9 *Experiment No. 9*

In this case the model is trained on dataset forrest 0.8 with 31 instances and 100% accuracy and testing is done on all the datasets, including forrest 0.8 also. From table 8, following points are made.

- Testing accuracy values of the model in this empirical study are 65.05%, 76.76%, 75.14%, 82.35%, 81.74%, 76.74%, 85.96%, 79.07%, 100%, 75.86%, 74.07%, 57.69% and 74.65% of Camel 1.6 to Combined Dataset, correspondingly.
- It is seen that the Nieruchomosci shows minimum accuracy of 57.69%.
- Dataset forrest 0.8 obviously is providing the maximum accuracy 100% when testing is done when the model is trained on the same dataset.
- The decreasing order of accuracy values during testing mode of datasets forrest 0.8, e-learning, jEdit4.3, Ivy 2.0, berek, Zuzel, Ant1.7, Combined Dataset, Intercafe, Camel 1.6 and Nieruchomosci are 100%, 85.96%, 82.35%, 81.74%, 79.07%, 75.86%, 75.14%, 74.65%, 74.07%, 65.05% and 57.69%, respectively.
- Datsets Tomcat 6.0 and arc provide, respectively, 76.76% and 67.74% accuracy of the model at testing stage, which are likely to be equal values.
- The average value of the accuracy in this case is 77.31%.

5.10 *Experiment No. 10*

In this experiment the model is trained on dataset Zuzel with 29 instances and 96.55% accuracy and tested an all the datasets starting from Camel 1.6 to Combined Dataset.

From table 8 it is found that the values of accuracy when applied on all the datasets under prediction varyes, and no specific pattern is shown by the model.

- It is noticeable that dataset Zuzel provides maximum accuracy of 96.55% during testing, when training is done on the same dataset, i.e., Zuzel.
- It is also seen that dataset Intercafe shows minimum accuracy during testing of 51.85% when training is done on Zuzel.
- Datasets Zuzel, berek, e-learning, forrest 0.8, Ivy 2.0, arc, Camel1.6, Ant1.7, Combined Dataset, jEdit4.3, Nieruchomosci, Tomcat6.0 and Intercafe provide accuracy of the model in decreasing order of 96.55%, 79.7%, 78.95%, 74.19%, 72.46%, 72.09%, 71.72%, 70.03%, 68.39%, 66.81%, 65.38%, 60.93% and 51.85%, respectively.
- On average, dataset Zuzel shows 71.42% accuracy.

5.11 *Experiment No. 11*

In this case the model is trained on dataset Intercafe with 27 instances and 100% accuracy and testing is done on all the datasets, including Intercafe also. From table 8 the following points are made.

- The prediction accuracies of the model from this empirical study are 49.77%, 75.63%, 55.25%, 52.52%, 62.03%, 73.49%, 64.91%, 74.42%, 32.26%, 44.83%, 100%, 76.92% and 59.72% of datasets Camel 1.6 to Combined Dataset, correspondingly.
- It is seen that the dataset forrest 0.8 shows minimum accuracy of 32.26%.
- Dataset Intercafe obviously provides the maximum accuracy of 100% when testing is done.
- The decreasing order of accuracy values of during testing mode of datasets Intercafe, Nieruchomosci, Tomcat 6.0, berek, arc, e-learning, Ivy 2.0, Combined Dataset, Ant1.7, jEdit4.3, Camel 1.6, Zuzel and forrest 0.8 are 100%, 76.92%, 75.63%, 74.42%, 73.49%, 64.91%, 62.03%, 59.72%, 55.25%, 52.52%, 49.77%, 44.83% and 32.26%, respectively.
- The average value of the accuracy in this case is 63.21%.

5.12 *Experiment No. 12*

In this experimental work, the model is trained on dataset Nieruchomosci with 26 instances and 100% accuracy and testing is done on all the datasets, including Nieruchomosci also. From table 8, following points are made.

- Accuracy of the model at testing is 72.96%, 57.54%, 68.65%, 43.07%, 59.71%, 80.00%, 78.95%, 86.05%, 77.42%, 75.86%, 55.56%, 100% and 64% of datasets Camel 1.6 to Combined Dataset, correspondingly.
- It is seen that the dataset jEdit 4.3 shows minimum accuracy of 43.07%.
- Dataset Nieruchomosci provides the maximum accuracy of 100% when both testing and training are done on it.
- The decreasing order of accuracy values during testing mode of datasets Nieruchomosci, berek, arc, e-learning, forrest0.8, Zuzel, Camel1.6, Ant1.7, Combined Dataset, ivy2.0, Tomcat6.0, Intercafe and jEdit4.3 is 100%, 86.05%, 80.00%, 78.95%, 77.42%, 75.86%, 72.96%, 68.65%, 64.00%, 59.71%, 57.54%, 55.56% and 43.07%, respectively.
- The average value of the accuracy in this case is 70.75%.

5.13 *Experiment No. 13*

In this case the model is trained on Combined Dataset with 3597 instances and 85.18% accuracy and testing is done on all the datasets, including datasets Camel1.6 to Nieruchomosci.

- It is noticeable that dataset jEdit4.3 provides maximum accuracy of 96.00% during testing when the model is trained on Combined Dataset.
- It is also seen from table 8 that dataset Zuzel provides minimum accuracy, i.e., 55.17%.
- Datasets Camel1.6 to Combined Dataset provide accuracy at testing stage of 79.3%, 90.7%, 79.42%, 96.00%, 89.28%, 87.91%, 91.23%, 74.42%, 93.55%, 55.17%, 85.19%, 61.54% and 85.18%, correspondingly.
- There is a decreasing trend of accuracy from datasets jEdit4.3, forrest0.8, e-learning, Tomcat6.0, ivy2.0, arc, Intercafe, Combined Dataset, Ant1.7, Camel1.6, berek, Nieruchomosci and Zuzel, which is 96.00%, 93.55%, 91.23%, 90.70%, 89.28%, 87.91%, 85.19%, 85.18%, 79.42%, 79.30%, 74.42%, 61.54% and 55.17%, respectively.
- The model provides on average 82.22% accuracy at testing on this dataset (Combined Dataset).

5.14 *Accuracy computation of the model in software bug prediction*

The proposed software bug prediction model (SBPS) is under predictive measures. For the prediction process first of all it is assumed that the bug value of any new system has to be tested and fixed to either all zeros or all ones. Then the proposed SBPS, which is already trained on validated Combined Dataset with Logistic Regression Classifier designed on 14 metrics set, is now applied for

prediction of the new system provided that the new system is compatible to the 14 metrics set; if this new system fails here then that missing metric value should be set to zero to make it compatible. This means that no data validation will be applied on this original dataset.

The bug prediction has used the proposed SBPS on datasets Tomcat 6.0, jEdit 4.3, ivy 2.0, System Data Management and Wspomanagie and predicted 96.62%—faulty, 3.38%—non-faulty for Tomcat 6.0 (tables 6 and 7), 96.54%—faulty and 3.46%—non-faulty for jEdit 4.3, 96.31%—faulty and 3.69%—non-faulty for ivy 2.0, 100% faulty and 0% non-faulty for System Data Management and 100% faulty and 0% non-faulty for Wspomanagie, showing the existence of modules with all the bug values zero or at all bug values one. It is also seen that those datasets that are not used during model construction like System Data Management and Wspomanagie also provide good result in terms of accuracy.

6. Conclusions

The goal of this paper is to investigate and assess the relationship between object-oriented metrics and defect prediction in terms of bug by computing the accuracy of the proposed model on different datasets.

It is concluded from all the experiments discussed in the paper that some datasets prompting similar prediction accuracy result and some give different results in terms of accuracy at prediction stage in one dataset. Similar conditions occur when training and testing both are done on all the datasets.

This type of variation in the accuracy at prediction (testing) stage may be due to different softwares (datasets) that have been selected in this study, among which all have been developed in different environments, by different staff members in terms of their expertise and skills, by different organizations. Are these organizations ISO certified or not? How frequently the staff members of these organizations switching to different companies? Under what conditions the SRS (Software Requirement Specification) has been prepared? Is the SRS complete, unambiguous, correct, valid, verifiable, traceable, testable, consistent and modifiable or not? And the most important one is design—is the design of the model in which the software (dataset) is developed valid? What about its level? Whether low level and high level design has been done prior to coding stage or not? These are the most important points on which the result will depend. If approximately all these conditions are fulfilled, then there will a less chance of occurrence of bug, and the quality of the software will improve.

Finally we saw that our proposed model provides on average 76.27% accuracy at testing (prediction) stage when training is done on all the datasets.

7. Future work

If some other real datasets are tested on the proposed model, this will provide the fault-prone classes that are to be tested carefully. The result provided by the model in terms of fault proneness towards a particular class will provide clear-cut guidelines to testers to work on those classes carefully. These guidelines will also save cost in reviewing the whole classes of the new system (software datasets).

The conclusions made on the basis of our experimental work shown in section 5 are encouraging from an experimental and practical point of view; this urges further studies to corroborate our findings and conclusions. More investigations are required to sketch any specific pattern.

References

[1] The Rotarian 1991 Rotary International, vol. 159, No.4, ISSN 0035-838X, https://books.google.co.in/books?id=eDIEAAAAMBAJ

[2] Singh Y, Kaur A and Malhotra R 2009 Software fault proneness prediction using support vector machine. In: *Proceedings of the World Congress on Engineering*, vol. 1

[3] Kan S H 2003 *Metrics and models in software engineering*, 2nd ed. Pearson Education

[4] Wellman B 1998 Doing it ourselves: the SPSS manual as sociology's most influential recent book. USA: University of Massachusetts Press, pp. 71–78, ISBN 978-1-55849-153-3

[5] Chidamber S R and Kemerer C F 1991 Towards a metric suite for object-oriented design. In: *OOPSLA '91 Conference Proceedings on Object-oriented Programming Systems, Languages, and Applications*, New York, USA: ACM

[6] Chidamber S R and Kemerer C F 1994 A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.* 20(6)

[7] Emam K E, Benlarbi S, Goel N and Rai S N 2001 The confounding effect of class size on the validity of object-oriented metrics. *IEEE Trans. Softw. Eng.* 27(7)

[8] Gursaran and Roy G 2001 On the applicability of Weyuker Property 9 to object-oriented structural inheritance complexity metrics. *IEEE Trans. Softw. Eng.* 27(4)

[9] Brito A F and Carapuca R 1994 Candidate metrics for object-oriented software within a taxonomy framework. *J. Syst. Softw.* 26: 87–96

[10] Jureczko M and Madeyski L 2006 A review of process metrics in defect prediction studies. In: *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA, USA, July 17–20

[11] Jin C, Jin S W, Ye J M and Zhang Q G 2009 Quality prediction model of object oriented software systems using computational intelligence. In: *Proceedings of the IEEE 2nd International Conference on power Electronics and Intelligent Transportation system*

[12] Jureczko M and Madeyski L 2010 Towards identifying software project clusters with regard to defect prediction. In: *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (Promise-10),

Timisoara, Romania. New York, USA: ACM http://doi.acm.org/10.1145/1868328.1868342
[13] Catal C 2011 Software fault prediction: a literature reviews and current trends. *Proc. Expert Syst. Appl*. 38: 4626–4636
[14] Okutan A and Yildz O T 2014 Software defect prediction using Bayesian Networks. *Empirical Softw. Eng*. 19: 154–181
[15] https://code.google.com/p/promisedata/w/list
[16] https://code.google.com/p/promisedata/wiki/MarianJureczko
[17] Henderson-Sellers B 1996 *Object-oriented metrics measures of complexity*. Prentice Hall
[18] Witten I H and Frank E 1999 *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann. http://www.cs.waikato.ac.nz/ml/weka/