March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

Object Oriented Design Metrics for Software Defect Prediction: An Empirical Study

Mrs. Bhagyashri Sunil Deshpande[1], Dr. Binod Kumar[2], Dr. Ajay Kumar[3]
[1]Resarch Scholar, Department of Computer Science SPPU, Pune, Maharashtra, India.
[2]SMIEEE, Director, JSPM's Jayawant Institute of Computer Applications, Pune, Maharashtra, India.
[3]Director, JSPM's Jayawant Technical Campus, Pune, Maharashtra, India.

Article Info
Volume 83
Page Number: 10501 - 10518
Publication Issue:
March - April 2020

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 13 April 2020*

Abstract

Reliability of software in respect to defect prediction is a trending topic of study. It is observed that many effective categorical models rely on requirement and design phase metrics and few consistent models include metrics from design phase. It is also noted that most studies show qualitative advantages by using Early Software Defect Prediction models. It is necessary to validate software metrics in error prevision for object-driven methods by means of statistical methods and machine learning. The quality of the software product in a software organization is assured by the validation process. Object-oriented metrics play a key role in fault estimation. This paper discusses the use of Chidamber and Kemerer (CK) metrics, QMOOD, Size, Complexity and Martin's metrics for the assessment of software defects. In this study bug/defect is treated as a dependent variable and the considered metric fit as independent variables. For defect detection two data analysis techniques, logistic regression and Decision Tree, are applied, validated and their statistical efficacy and performance measures are discussed.

Keywords: *Software Reliability, Object-Oriented Software Metrics, Software bug prediction, Machine Learning, Logistic Regression, Decision Tree.*

I. INTRODUCTION

The present development of software is largely based on an object-oriented paradigm. Software metrics can best be used to assess the quality of object-oriented software. Under the IEEE software engineering standard, object design in the software development environment is becoming more important, and software measurements are essential for the quantification of the software quality in the software engineering business sector. These metrics help to check the quality of software properties such as fault detection. The significance of these metrics lies in their ability to predict the software's reliability. Software performance is generally referred to in practice as reliability, maintenance and understanding. The number of faults in the developed software typically tests reliability.

We find that earlier there were studies undertaken for detecting defect proneness [3, 6-8, 10, 11]. In these studies, software metrics for mathematical models, to forecast fault inclination,are used [12, 13, 16, 18, 19]. Further, empirical analysis of the results from machine learning methods is essential for understanding the effect of these matrics used for prediction and model development.The information obtained from these empirical researches is believed to be the main support to design and develop the models. The capability to assess the process was widely accepted as an important aspect of quality management. With their key role in the design and implementation of computer information systems, managers are constantly aiming to improve application development processes.The software quality is the principal criteria for a client, academics software organizations and researchers. “Quality is not accidental. But it is a result of a wise and intelligent effort"[1]. Defects of software can only be predicted on the basis of historical data collected in the implementation or data acquired during design processes in the software development process. The object-oriented approach is different from the traditional approach to programming. It is used to distinguish information and control, based on objects, each of which includes a set of specified data and a set of fixed predefined operations known as data-driven methods. The OO approach offers greater reusability, reliability and maintainability than the conventional, functional decomposition-based approach. Encapsulation is the attribute of OO System that creates objects which are easy to integrate into a new design, and which are essentially conducive to reusability [2]. The significance of various software metrics for defect prediction has been investigated in this study. The findings of building defect prediction models have been analyzed. The study analyzed that parameters were selected and filtered for the model. In conclusion, the results were compared and the most promising method for predicting defects were identified and recommended. The paper starts with an insight into the related work in the field, which study the methodologiesand evaluation of Software Reliability.It furtherexplains models developed during research with their results and conclusion parameters from it. The paper finally provides with a conclusion and future scope in the study.

II.Research Objective

Reliable software is essential for business processes that are complex in nature. Designing such software is a main challenge faced in software industry today. Various techniques have been used for predicting and estimating the error prone classes of the software. The primary goal of the research study is to develop an assessment model for software reliability by early defect prediction. Using Machine Learning (ML) algorithms, this research is intended to empirically predict the software defect prone areas or attributes.

III.Related Work

The quality of software has long been studied. The literature contains a number of software quality models. In recent decades, numerous software developers and researchers have carried out software quality prediction studies.In this fieldof detecting the software reliability, neural

networks, fuzzy logic, regression tree, etc. and their techniques were used.

Chidamber and Kemerer [2] discussed their research work on OO metrics at the software development cycle at design stage. The metric calculations were based on theory and are used by well-trained OO developersduring development. Chidamber and Kemerer [2] emphasize on the main measurement criteria to improve software quality with a new suite comprising six design levels metrics called WMC, DIT, NOC, CBO, RFC and LCOM.

In order to determine the overall impact of class sizes on the validation of OO metrics, Emam et al[6] used Chidamber and Kemerer metrics suite and Lorenz and Kidd sub-set metrics. Their primary consent in the paper was to estimate the proneness of class defects. The researchers have shown that the size of the product metrics has a confounding effect on fault-proneness. An observational approach to determine if there is any adverse effect has been experimentally tested and justified. It has eventually carried out a study with a large C.

In the Chidamber and Kemerer metrics Gursaran and Roy [13] found that the property of Weyuker 9 is not satisfied by any inheritance metric. Inherited metrics proposed by Brito and Carapuca [5], the authors showed that a particular class of structural inheritance metrics, defined on an abstract of the legacy structure, can never accomplish Weyuker's 9 property, building on the metric assumptions and concatenation definitions given by Chidamber and Kemerer. An analysis focused on processes e.g. the number of modifications, number of newly revised lines and number of defects, was given by Jureczko and Madeysky [18]. They analysed and reported that all of the above metrics rely on data for changes in artefacts in the source code. It was observed that some of the changes may not have a direct connection with the software process. Finally, the researchers showed that system metrics can be an effective addition to prediction modules that are usually based on prediction methods. Jin et.al [29] have suggested, developing a fault proneness prediction model for Fuzzy c-means clustering (FCM) for clustering, and radial base function neural network (RBFNN) for classification. The analysis of a prediction of defects by Jureczko and Madeyski [18] for software projects shows that they have used 92 versions of open source projects and 38 proprietaries on the data repository.The classes of similar projects have been used in this paper to classify Hierarchic and K-mean clustering and Kohonen's neural networks. For each of the identified classes, two defect prediction models were created. The authors have identified two clusters and compared results obtained by other researchers. Finally, that concluded to clusteringbeing suggested and applied for further use.

Catal [16] reported the survey of 90 research papers in concern with the research. The numbers of previous changes in a file showed Graves et al. [7] to be a strong fault forecast. In particular, they found that when the number of times a module is changed, the number of lines of code in a module is not helpful in predicting defects. The authors found that the context of the change provides more

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

useful information than the measurements of size and structure could be collected. Moser et al [17] performed a comparative analysis of the process efficiency and defect prevention product metrics and showed a great improvement in the process metrics.

IV.RESEARCH BACKGROUND

This section encapsulatesrealistic data collection correlated with assigned independent and dependent variables.

A. Data Collection

For this research, dataset is collected for the experimental work, which is considered from the projects for OO-deficiency of the Marian Jureczko data sets [31] available at http://madeyski.e-informatyka.pl/tools/software -defect-prediction/ it consists of study of 13 open source projects of similar kind having total instances 46,775. Each instance of the dataset has corresponding metrics sets with associated bug values. The more the data sample better the accuracy of the model.

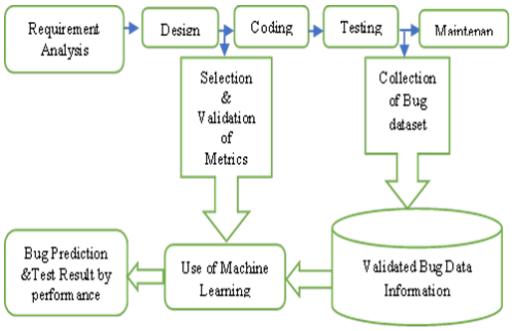

Figure 1: Architecture of Performance Evaluation During Design Phase of SDLC

B. Dependent and Independent Variables

In this research bug is dependent variable and the independent variables are CK suite(WMC, DIT, NOC, CBO, RFC LCOM)+Martins metrics(CA, CE)+ LOC+QMOOD Suite(LCOM3, NPM, DAM, MOA, MFA, CAM)+Extended CK(IC, CBM, AMC)+McCabe's(MAX_CC and AVG-CC).

V.METHODOLOGY ADOPTED

This research work, explores different metrics for predicting bugs using the ML techniques Decision Tree (DT) and Logistic Regression (LR). The entire data-set splitting is done into two parts, 70:30 ratio as training and test data set. The training data set is applied for prediction using ML models. Testing dataset is applied to validate the prediction model for software reliability.

A. Metrics Description

The study consisted of 20 independent OO metrics summarized in table 1.

Table 1: Metrics Description

Suite 1	CK Metric (6 metrics)
WMC	Weighted Methods per Class
DIT	Depth of Inheritance tree
NOC	Number of Children
CBO	Coupling Between Objects increased as the methods of one class access

	services of another.
RFC	Response for Class, number of methods invoked in response to message to the object.
LCOM	Lack of cohesion in methods, i.e. the number of pairs of methods not sharing a reference to an instance variable.
Suite 2	Martins metric (2 metrics)
CA	Afferent Coupling, how many other classes use the particular class.
CE	Efferent Coupling, the no of classes is used by the particular class.
Suite 3	QMOOD (5)
NPM	Number of Public Methods
DAM	Data Access Metric, ratio of private (protected) attributes to total attributes
MOA	Measure of Aggregation, count of the number of data declarations (class fields) whose types are user defined classes
MFA	Measure of Functional Abstraction, number of methods inherited by a class plus number of methods accessible by member methods of the class
CAM	Cohesion Among Methods, summation of no of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods.
Suite 4	Extended CK suite (4 metrics)
LCOM3	Lack of cohesion in methods 3
IC	Inheritance coupling, Number of parent classes (the number of inherited methods and variables) with which a given class is associated
CBM	total number of new/reddened methods to which all the inherited methods are coupled
AMC	Average methods per class
Suite 5	
LOC	Lines of code
Suite 6	McCabe's Cyclomatic Complexity (2 metrics)
MAX CC	maximum McCabe's cyclomatic complexity
AVG CC	Average McCabe's cyclomatic complexity
defect/ bug:	Defects found in post-release bug-tracking systems.

B. Metrics and Model Assumption

Aim is to quantify the impact of OO design metric on software quality i.e. Reliability by building Predictive models. In this study we want to show how all considered metrics are significantly contributing in model building. For software bug prediction there are certain assumptions about metrics that are described below.

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

a. The model is built by OO metrics.

b. The analyzed model consists of 20 OO metrics: WMC, DIT, CBO, RFC, COM, CA, CE, NPM, COM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX-CC and AVG CC.

c. In the absence of metric value it is considered to be zero.

d. Dependent variable is 0 or 1 as bug value for every instance.

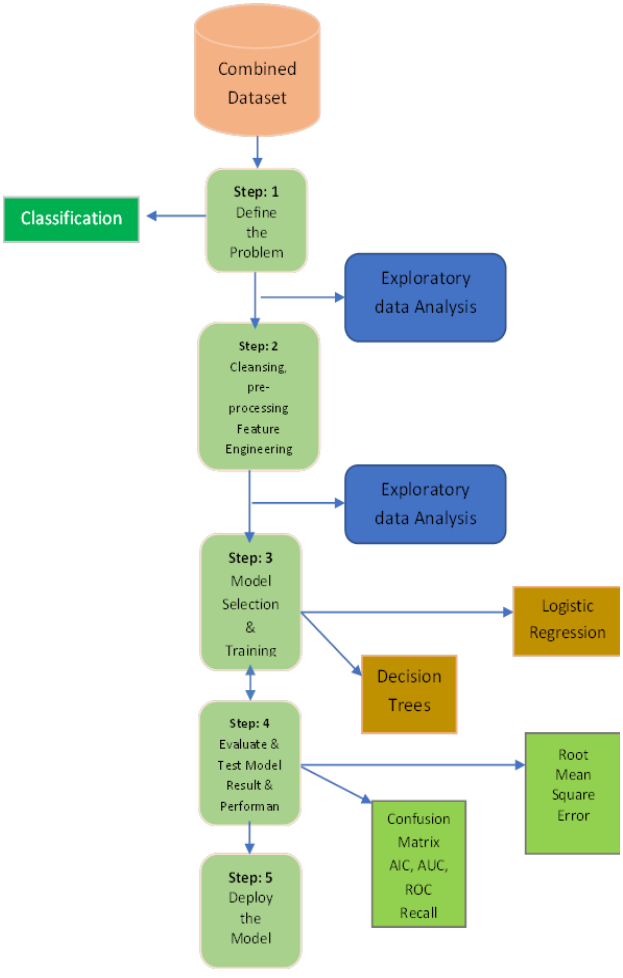

Figure 2: Framework for Modeling and Assessment

VI.MACHINE LEARNING TECHNIQUES

Machine learning is a kind of artificial intelligence that enables computers to learn without explicit programming. Machine learning is based on software development programs which can grow and learn in the context of new data. Because of better results researchers will be using machine learning which can also deals with complex data as well.

A. *Logistic regression analysis*:

Logistic Regression is a classification algorithm. It is used to predict a binary outcome (0=Not Bug, 1=Bug). The logistic regression is a predictive analysis similar to the linear regression models. Logistic Regression plays vital role in describing data and relationship between one dependent (binary variable) and one or more independent variables (continuous-level/ interval / ratio scale). In simple words, by applying data to a logit equation, it estimates the likelihood of an event occurring. glm (formula = bugs ~., family = binomial(logit), data = bugtrain1)

Generalized linear model (GLM) –output variables with error distribution models other than normal distribution are allowed by this multipurpose extensive linear regression model. The general linear model is a regression model for statistical analysis. $d(y, y) = 0, d(y, \mu) > 0 \; \forall \; y \neq \mu$ For mapping of predicted values to probabilities there is a use of Sigmoid function. This function maps the value between 0 and 1 of any real value. In machine learning generally we make use of sigmoid to map predictions to probabilities.

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

$f(x) = \frac{1}{e^{-x}}$----------(1)

For logistic regression it is

$\sigma(Z) = \sigma(\beta_0 + \beta_1 X)$ ----------(2)

We have expected that our results will give values between 0 and 1.

$Z = \beta_0 + \beta_1 X$ ----------(3)

$h_\theta(x) = \text{sigmoid}(Z)$

i.e. $h_\theta(x) = \frac{1}{1+e^{-(\beta 0+\beta 1 x)}}$---------- (4)

B. *Decision Tree*:

A decision tree is a decision support and a popular tool in machine learning. Decision tree can be a type of supervised learning algorithm with a pre-defined output variable that isnormally utilized in classification problems.It works for the input and output variables both categorical and continuous.This technique, based on most significant splitter / differentiator in input variables, the population or sample is split into two or more homogeneous sets (or sub-populations). Decision Tree Analysis (0=Not Bug,1=Bug)

VII. Proposed Algorithm and Framework

Algorithm for Model Development

1. Input: A set of datasets VDS(validated Combined Dataset)
2. a set of metrics (MT = MT1, MT2, MT3,…,MTn);
3. A set of classifiers (CL1, CL2, CL3…, CLz).
4. Output: Proposed model based on error (RMSE).

Algorithm steps:

1. Select datasets of similar kind of project.
2. Combine all the datasets to form validated dataset VDS=∑D1+D2+D3….Dm.
3. Identify and remove Near Zero variance metric.
4. Check the missing values in each metrics. Replace by measures of central tendency.
5. Check the outliers and treat.
6. Split data into Train and Test (70:30).
7. Apply different classifiers CL1,CL2,...,CLz) using Train Dataset & Validate using Test Dataset
8. Build Model with significant metrics MT1 …MTn
9. Generate Confusion matrix
10. Apply Performance Measure calculate, Accuracy, Recall, AUC, ROC for each classifier on VDS
11. Select foremost Classifier with Maximum Accuracy and minimum RMSE.

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

VIII. STATISTICAL EFFICACY MEASURES

In this study, the distribution mean, median, and interquartile ranges of each metric are studied. The distribution of a metric determines the applicability of following regression analysis techniques. Descriptive statistics of metrics on related data is represented in Table 2.

Table 2: Descriptive Statistics of Metrics

Metric	Min	1st Qu	Median	Mean	3rd Qu.	Max.
WMC	0	2	4	6.33	7	212
DIT	0	1	2	2.49	3	9
NOC	0	0	0	0.49	0	467
CBO	0	4	9	13.41	17	860
RFC	0	7	16	25.19	31	428
LCOM	0	1	1	41.75	10	11323
CA	0	0	1	3.54	2	860
CE	0	2	7	10.10	15	133
NPM	0	2	2	4.55	4	212
LCOM3	0	0.85	1	1.26	2	2
LOC	0	28	76	210.70	196	13175
DAM	0	0	0	0.29	0.6	1
MOA	0	0	0	0.29	0	158
MFA	0	0	0.75	0.50	0.89	1
CAM	0	0.37	0.57	0.53	0.67	1
IC	0	0	0	0.77	1	6
CBM	0	0	0	1.40	2	33
AMC	0	7.5	17.33	30.81	37.29	3492
MAX.CC.	0	1	1	3.70	4	252
AVG.CC.	0	0.5	1	1.343	1.6	30.5

IX.MODEL BUILDING

The Framework for modelling and assessment is shown in figure 2. It present,Architecture of Data Validation, Metric selection, Classifier selection,block diagrams of Data Pre-processing and Accuracy calculation using the proposed model. The model is build using all metric by considering the formula of logistic regression.

Pred bug=1/1+e^- [(2.92E-02)*WMC + (-1.52E-01)*DIT + (1.76E-02)*CBO + (-1.62E-03)*RFC + (-6.96E-04)*LCOM + (-1.09E-02)*CA + (2.06E-02)*CE + (-2.61E-05)LOC + (4.19E-01)*LCOM3 + 1.38E-02*NPM + (-7.27E-02)*DAM + (7.45E-02)*MOA+ (8.25E-01)*MFA + (-1.30E+00)*CAM + (-3.04E-02)*IC + (5.86E-02)*CBM +(6.58E-03)*AMC + (-8.75E-03)*MAX_CC+(3.78E-02)*AVG-CC]---------- (5)

Derived from this glm (formula = bugs ~., family = binomial (logit), data = bugtrain1)

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

Table 3: Coefficient table output

Coefficients	Estimate	Std. Error	z value	Pr(>|z|)	Signifi-cant Rate
(Intercept)	-1.83E+00	7.10E-02	-25.768	< 2e-16	***
WMC	2.92E-02	6.33E-03	4.612	3.99E-06	***
DIT	-1.52E-01	1.86E-02	-8.17	3.10E-16	***
CBO	1.76E-02	1.36E-02	1.295	0.1953	
RFC	-1.62E-03	1.12E-03	-1.45	0.1471	
LCOM	-6.96E-04	8.25E-05	-8.434	< 2e-16	***
CA	-1.09E-02	1.35E-02	-0.802	0.4223	
CE	2.06E-02	1.36E-02	1.509	0.1313	
NPM	1.38E-02	5.39E-03	2.552	0.0107	*
LCOM3	4.19E-01	2.88E-02	14.553	< 2e-16	***
LOC	-2.61E-05	6.59E-05	-0.397	0.6916	
DAM	-7.27E-02	4.40E-02	-1.653	0.0983	.
MOA	7.45E-02	1.42E-02	5.245	1.56E-07	***
MFA	8.25E-01	6.24E-02	13.22	< 2e-16	***
CAM	-1.30E+00	6.89E-02	-18.851	< 2e-16	***
IC	-3.04E-02	2.79E-02	-1.092	0.275	
CBM	5.86E-02	8.21E-03	7.14	9.35E-13	***
AMC	6.58E-03	4.58E-04	14.371	< 2e-16	***
MAX.CC.	-8.75E-03	3.62E-03	-2.417	0.0157	*
AVG.CC.	3.78E-02	1.51E-02	2.5	0.0124	*

[Signif.codes:0 '***'0.001'**' 0.01 '*' 0.05 '.' 0.1 ' ' 1] the table 3 defines the relation between the regression coefficients and values of intercept, standard error, z-value and p-value. On various parameters for each coefficient: the intercept and other independent variables. (*** – high importance, * – medium importance, and dot – next level of importance). In this context for significant variables estimate, Std. Error, Z-value, Pr (>|z|) can be observed from table 3. It can be seen that WMC, DIT, LCOM, LCOM3, MOA, MFA, CAM, CBM, AMC, shown as *** as highest significance rate.* least significanceis NPM, MAX_CC and AVG-CC and lastly seen as. (dot) dam very low significant whereas CBO, RFC, CA, CE, LOC, IC has no significance in model building.

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

X. Performance Measures and discussion

Experiments were conducted in R tool for Machine learning techniques and build the model using Logistic Regression, Decision Tree. Further to assess the performance of a logistic regression model, we must study few measures.

A. Experimental analysis of Logistic Regression

1] Null and Residual Deviance

Instatistical hypothesis testing deviance is a quality-of-fit measurement for a model that is regularly used.Where model-fitting is achieved by maximum likelihoodby a generality of the impression of using the sum of squares of residuals in ordinary least squares to cases, it plays an vital role in exponential dispersion models and generalized linear models.

The unit deviance d(y, μ) ----------(6)

is a bivariate function that satisfies the following conditions:

D(y,y)=0 and D(y,μ)= ∑ d(yi,μi) . The (total) deviance for a model M0 with estimates

μ= E [Y/θ_0] ----------(7)based on a dataset y, may be constructed by its likelihood.

Null Deviance: - Deviance is a fitness measure of a model. Higher numbers usually explain the poor fit. The null deviance indicates how well a model predicts the response variable, which contains only the intercept (great mean) while the residual one with independent variables is included.

From the model summary, the response bug variable is affected by WMC, DIT, NOC, CBO, RFC LCOM, CA, CE, LOC, LCOM3, NPM, DAM, MOA, MFA, CAM, IC, CBM, AMC, MAX_CC and AVG-CC variables.The legend of the correlated coefficients (* * * – high priority, * – medium importance and dot – next level importance) realizes the rank of the variable. Rerunning the model with these significant independent variables will impact the model performance and accuracy.

Reduced Model of Logistic Regression is built with significant Variable:

glm (formula = bugs ~ wmc + dit + lcom + npm + lcom3 + dam + moa + mfa + cam + cbm + amc + max.cc. + avg.cc., family = binomial(logit), data = bugtrain1) ---- (8)

It is observed that the Null deviance and degree of freedom of full and reduced model is 38197 on 32743 degrees of freedom and Residual deviance as 34429 on 32724 degrees of freedom for full model and 35037 on 32730 degrees of freedom. Deviance is a measure of goodness of fit of a model. Higher numbers always indicate bad fit. The null deviance shows how well the response variable is predicted by a model that includes only the intercept (grand mean) whereas residual with inclusion of independent variables.

The reduced model only the significant variables that are seen by degree of freedom (i.e 32743-32730=13). Model only with significant variables increases the deviance to 35037 from 34429; it's a significant increase in devianceIf you have a very small Null Deviance, the Null

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

Model explains the data fairly well. With your Residual Deviance, too.

2] AIC (Akaike Information Criteria) –

The analogous metric of adjusted R^2 in logistic regression is AIC. AIC is the fitness measure which penalizes the number of model coefficients in the sample.

Table 4 : AIC Values of LR Model

Model	AIC:
Logistic Full Model	34469
Logistic Reduced Model	35065

In full model of logistic regression all metrics are contributing to the model development where as in reduced model only significant variables. Significant variables are derived from P-values for regression coefficients. Each regression coefficient in reduced model is statistically significant(P-value<0.05). As reduced model has AIC value high as compared with AIC value of full model observed in table 4, hence Full model is better than reduced model.

3] Confusion Matrix

The confusion matrix is a contingency table of 2X2 dimension, a binary classification problem with two possible classes Positive and negative. In this study classes having bug and not bug, the consideration is that the not bug is positive case and the bug is negative case

4] Model Evaluation

The classification model performance evaluation is done by performance metrics on the basis of confusion matrix. A binary classifier can make two possible errors: false positives (FP) and false negatives (FN). In addition, a correctly classified buggy class is a true positive (TP) and a correctly classified non-buggy class is a true negative (TN). We have evaluated results of binary classification results given as follows:

1. Total no of instances = properly classified instances + wrongly classified instance---------- (9)

2. properly classified instance = TP +TN

3. wrongly classified instance = FP+FN

4. Accuracy: How often the classifier is correct.accuracy = (TP + TN)/ (TP + TN + FP + FN)----(10) i.e accuracy = properly classified instances / no of instances.

5. Misclassification Rate: How often is it wrong. Misclassification rate=1-Accuracy i.e Error Rate =(FP + FN)/(TP + TN + FP + FN) ----------(11)

6. True Positive Rate(Sensitivity /Recall): when its actually yes and how often does it predicts yesSensitivity=TP/FN+TP----------(12)

7. False Positive Rate: when its actually No and how often does it predicts yes: FP/FP+TN----------(13)

8. Specificity : When it's actually No, how often does it predict No : Specificity =TN/TN+FP ---------- (14)

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

9. Precision : When it predicts yes, how often is it correct

Precision = TP/TP+FP----------(15)

10. Prevalence: how often does the yes condition actually occur in our sample.

Prevalence = FN+TP/(TP + TN + FP + FN)---------(16)

i] LR Train Data

As per the values in Table 5 accuracy is calculated with properly classified instances by dividing total no of instances whose rows correspond to the observed values and the predicted values by the classification model are shown at the columns

Table 5 : Confusion Matrix for Train Data

n=32744	Pred-N	Pred-Y
Actual-N	22944	7230
Actual-Y	959	1611

The model classifies the classes with overall 74.99% accuracy and misclassification with 25.01%.

Table 6: Performance Measures of LR Train Data

Accuracy	74.99%
Misclassification Rate:	25.01%
Sensitivity	95.99%
Specificity	18.22%
PosPred Value	76.04%
NegPred Value	62.68%
Prevalence	73.00%
CI (0.7452, 0.7546)	95%
Area under the curve:	0.6936

Model also shows the positive predicted value as 76.04% and negative predicted value 62.68%. Model shows the 95.99% sensitivity and the specificity with 18.22%. The model also shows the 73% prevalence in the sample data. The different parameters observed and noted in the table are scaled in the graphical format in Figure 5.

5] Receiver Operating Characteristic (ROC)

Précises the model's performance by assessing the trade-offs between true positive rate (sensitivity) and false positive rate (1- specificity). To plot ROC, it is recommended to assume p > 0.5 since we are more concerned about success rate. ROC explains the predictive power for all possible values of p > 0.5.

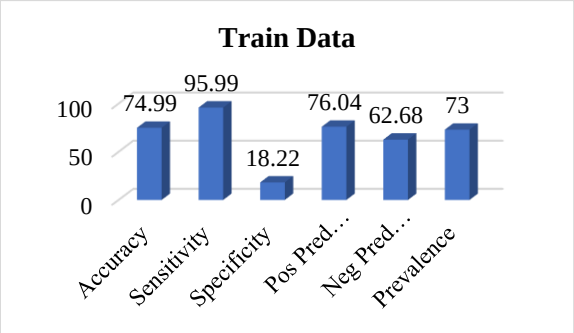

Figure 3 : Graph for Measures on Train Data

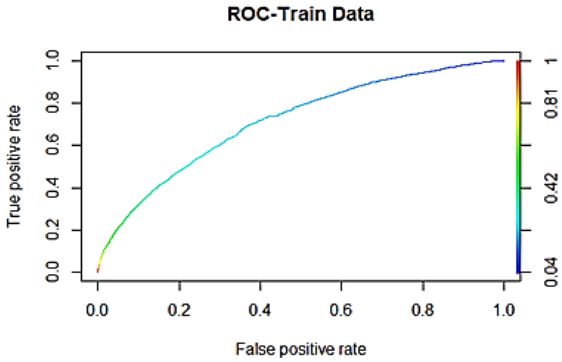

Figure4: ROC Curve for Train Data

The area under curve (AUC), referred to as index of accuracy (A), is a perfect

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

performance metric for ROC curve. Here is the sample ROC, higher area under curve means better prediction power of the model.The ROC of a perfect predictive model has TP (True positive) equals 1 and FP (False positive) equals 0. This curve will touch the top corner of the graph. ROC curve shows that AUC for train data is 0.6936

ii] Test Data

As per the confusion matrix of test model, during testing of model it can be seen that overall 75 % accuracy in classifying the classes and it misses to classify by 25%.

Table 7: Confusion Matrix Test Data

n= 14031	Pred N	Pred Y
Actual N	9818	3083
Actual Y	425	705

Table 8 : Performance Measures of Test Data

Accuracy	75.00%
Misclassification rate	25.00%
Sensitivity	95.85%
Specificity	18.61%
PosPred Value	76.10%
NegPred Value	62.39%
Prevalence	73.00%
CI (0.7427, 0.7571)	95%
Area under the curve:	0.6936

The model Correctly predicts the 76.10% positive prediction condition and there is 62.39% of negative prediction status. The sensitivity is 95.85% whereas specificity is 18.61%. The prevalence is observed as 73%.

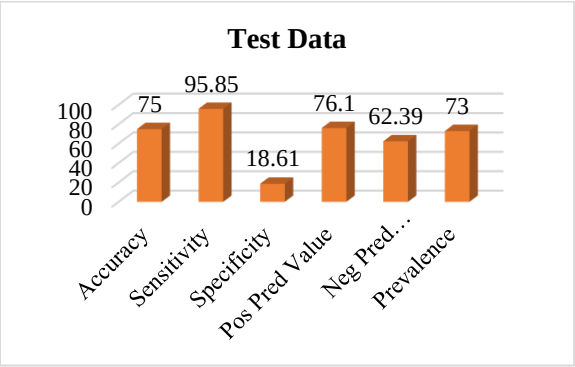

Figure 5 : Graph for Measures on Test Data

The graphical presentation of all the measures accuracy, specificity, prevalence, sensitivity positive predicted value and negative predicted value can on the graph in figure 6. There is 95% confidence interval ranges between 0.7452 and 0.7546 occurs in the considered data sample. The area under curve for test data is 0.6925. Hence model working good in both data sets. By using these measures, the selection of the best model is determined.

The model build on train data is tested and validated by test data which is graphically shown in figure 7. The graph shows the comparative analysis of both. There are perfect.Balancing results are observed through the various measures. It shows that the model fits well on the considered data.

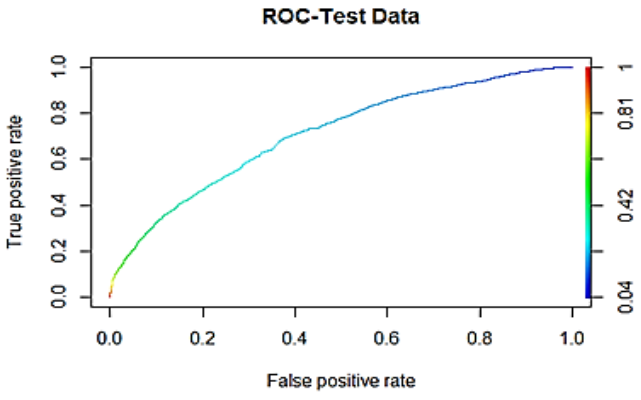

Figure 6 : ROC Curve for Test Data

March - April 2020
ISSN: 0193-4120 Page No. 10501 - 10518

Table No 9 : Comparative Analysis of Performance measures of Logistic regression

Model		**Train data**	**Test data**
Logistic Regression	Accuracy	74.99	75.00
	Misclassification	25.01	25.00
	Sensitivity	95.99	95.85
	Specificity	18.22	18.61
	PosPred Value	76.04	76.10
	NegPred Value	62.68	62.39
	Prevalence	73.00	73.00

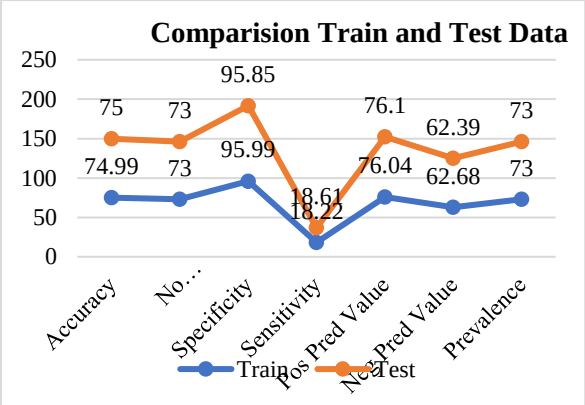

Figure 7 : Graph for Comparison Train & Test

With the choice of logistic regression, it is evident that the accuracy for this model is evaluated as 75% and error rate as 25%. The model accuracy improvement can be done with other classification models.

B. Experimental analysis of Decision Tree

Let's use the combined dataset and apply decision tree method to predict whether a class is buggy or not. The attributes are available to predict the Model. Below the data is split as training and test data, which will be used for building the model and predictions. Let's predict on fitted data and calculate misclassification percentage.

Table 10 : Confusion Matrix for Train Data

n= 32744	Pred N	Pred Y
Actual N	23423	7887
Actual Y	480	954

Model trained by train data set shows Overall 74.45% accuracy and it misclassify the classes by 25.55%. The model sensitivity is 97.99% and model specificity is 10.79%.

Table11 : Performance Measures of DT Train Data

Accuracy	74.45%
Misclassification Rate	25.55%
Sensitivity	97.99%
Specificity	10.79%
PosPred Value	74.81%
NegPred Value	66.53%
Prevalence	73.00%
CI (0.7397, 0.7492)	95%
AUC	0.6152379

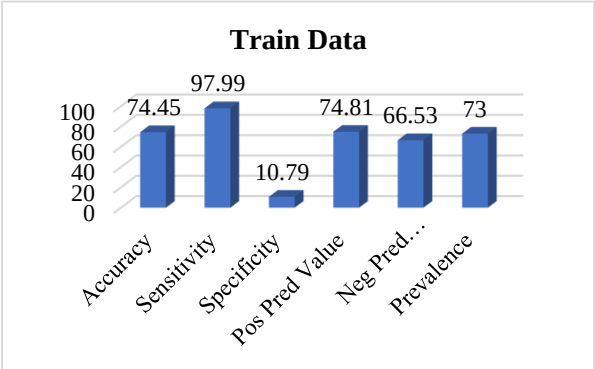

Figure 8 : Graph for Measures on Train Data

The positive predicted value is 74.81% and negative predicted value is 66.53. Prevalence can be seen that the 73% The Model shows the confidence interval of 95% in the range of 0.7397 and 0.7492 condition for the sample data used. The

positive predicted value is 74.81% and negative predicted value is 66.53%.

Figure 9 : ROC curve for train data

As per the developed model of decision tree the Confusion Matrix for Test Data is generated as:

Table 12: Confusion Matrix for Test Data

n=14031	Pred N	Pred Y
Actual N	10041	3363
Actual Y	202	425

The confusion matrix as above is for test data with 14031 total observations. It shows the overall measures for the total test data with the 95% confidence interval in the range of 0.7386 to 0.7531 and 73% prevalence.

The model is tested and validated on test data set. It shows overall 74.59% accurately classify the classes and overall 25.41% wrongly classify the classes by model.

Table 13 : Performance Measures of DecisionTree Test Data

Accuracy	74.59%
Misclassification Rate	25.41%
Sensitivity	98.03%
Specificity	11.22%
PosPred Value	74.91%
NegPred Value	67.78%
Prevalence	73.00%
CI (0.7386, 0.7531)	95%
AUC	0.609539

Positive predicted value is the 74.91% and negative predicted condition is 67.78%. The sensitivity is 98.03% and the specificity is

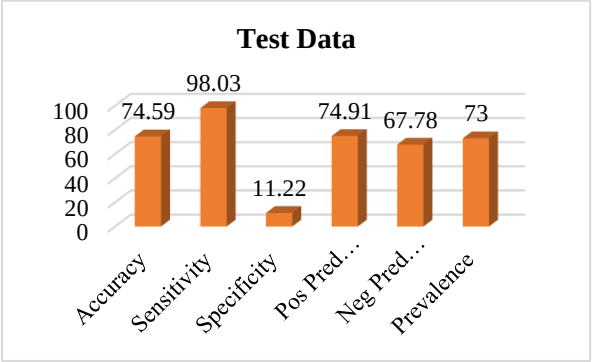

Figure 10 : Graph for Measures on Test Data

11.22% with the 73% prevalence actually occurs in sample data used.

Figure 11 : ROC curve for test data

Table 14: Comparative study of Performance Measure for Decision Tree

Models		Train data	Test data
Decision Tree	Accuracy:	74.45%	74.59%
	Misclassification Rate:	25.01%	25.41%
	Sensitivity	97.99%	98.03%
	Specificity	10.79%	11.22%

	PosPred Value	74.81%	74.91%
	NegPred Value	66.53%	67.78%
	Prevalence	73.00%	73.00%

The evaluation process is implemented using two real testing/debugging datasets. Experimental results are collected based on accuracy, precision, recall, measures. Results reveal that the Machine Learning techniques are efficient approaches to predict the future software bugs..

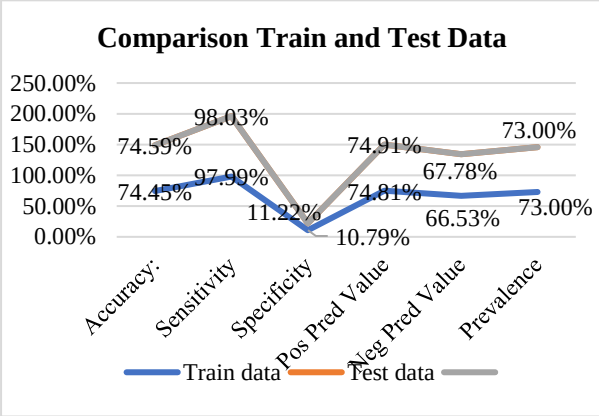

Figure 12 :Comparison Train and Test Data

The comparison results shows that the Logistic Regression classifier has the little better results over the Decision Tree. Moreover, experimental results showed that using Machine Learning approach provides a better performance for the prediction model

XI.CONCLUSION

The objective of this study is to inspect and assess the relationship between object-oriented metrics and defect prediction in terms of bug by computing the accuracy of the proposed model on combined datasets. The performance of Logistic regression and Decision Tree Machine Learning techniques has been estimated. The experiment for bug prediction using combined dataset of 13 software. We have empirically proved that defects predicted using mentioned ML techniques signifies that the actual and the predicted values are very close for all the data being taken into consideration in predicting reliability in terms of the various statistical efficacy measures applied. Two techniques the Logistic regression and Decision Tree shows the promising results with the accuracy of 75% and 74.5% respectively.

XII.FUTURE WORK

As a future work there is inclusion of other Machine Learning techniques to provide an extensive comparison among them. Furthermore, study of software metrics in the learning process is one possible approach to increase the accuracy of the prediction model in terms of software reliability.

REFERENCES

[1] Dr. Linda H. Rosenberg, Lawrence E. Hyatt "Software Quality Metrics for Object-Oriented Environments"

[2] Shyam R. Chidamber, Chris F. Kemerer "A Metrics Suite For Object Oriented Design" M.I.T. Sloan School of Management , Revised December 1993.

[3] SanjeevPunia, "A Review of Software Quality Metrics for Object-Oriented Design " International Journal of Advanced Research in Computer Science and Software Engineering , Volume 6, Issue 8, August 2016 ISSN: 2277 128X, www.ijarcsse.com

[4] R. A. Khan, K. Mustafa and S. I. Ahson "An Empirical Validation of Object Oriented Design Quality Metrics" J. King Saud Univ., Vol. 19, Comp. & Info. Sci., pp. 1-16, Riyadh (1427H./2007)

[5] Fernando Brito e Abreu, WalcelioMelo " Evaluating the Impact of Object-Oriented Design on Software Quality" https://www.researchgate.net/ publication/2818243

[6] Surbhi Gaur, Savleen Kaur, Inderpreet Kaur "Validation of Software Quality Models using Machine Learning: An Empirical Study" International Journal of Computer Applications (0975 – 8887, 08th National Conference on Next generation Computing Technologies & Applications (NGCTA - 2013)

[7] "Evaluating the Impact of Object-Oriented Design on Software Quality"

[8] The Software Design Metrics tool for the UML

[9] Dinesh Verma and Shishir Kumar , "An Improved Approach for Reduction of Defect Density Using Optimal Module Sizes, Hindawi Publishing Corporation Advances in Software Engineering Volume 2014, Article ID 803530, 7 pages http://dx.doi.org/10.1155/2014/803530

[10] KamrunNaharNeela*, Syed Ali Asif, Amit Seal Ami, AlimUlGias "Modeling Software Defects as Anomalies: A Case Study on Promise Repository" Journal of Software Volume 12, Number 10, October 2017

[11] Johny Antony P & Harsh Dev "Estimating Reliability Of Software System Using Object-Oriented Metrics" International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) ISSN 2249-6831 Vol. 3, Issue 2, Jun 2013, 283-294 © TJPRC Pvt. Ltd.

[12] Marian Jureczko, "Significance of Different Software Metrics in Defect Prediction"

[13] DharmendraLal Gupta, And KavitaSaxena, Software bug prediction using object-oriented metrics Sadhana Vol. 42, No. 5, May 2017, pp. 655–669 Indian Academy of Sciences DOI 10.1007/s12046-017-0629-5

[14] AnkushVesra, Rahul " A Study of Various Static and Dynamic Metrics for Open Source Software" International Journal of Computer Applications (0975 – 8887) Volume 122 – No.10, July 2015.

[15] Hemlata Sharma, Anuradha Chug "Dynamic Metrics are Superior than Static Metrics in

[16] Maintainability Prediction : An Empirical Case Study" 978-1-4673-7231-2/15/$31.00 ©2015 IEEE

[17] Shamsul Huda, Sultan Alyahya, MdMohsin Ali, Shafiq Ahmad, "A Framework for Software Defect Prediction and Metric Selection" IEEE Access Digital Object Identifier 10.1109/ACCESS.2017.2785445

[18] N.Kalaivani1, Dr.R.Beena " Overview of Software Defect Prediction using Machine Learning Algorithms" International Journal of Pure and Applied Mathematics Volume 118 No. 20 2018, 3863-3873 ISSN:1314-3395 (on-line version) url: http://www.ijpam.eu

[19] Manjula. C.M. Prasad, Lilly Florence, and Arti Arya A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques" International Journal of Database Theory and Application Vol.8, No.3 (2015), pp.179-190 http://dx.doi.org/10.14257/ijdta.2015.8.3.15

[20] AynurAbdurazik and XiaochenShen "ATool for Measuring Java OO Couplings"

[21] Shweta Sharma, Dr S. Srinivasan " A review of Coupling and Cohesion metrics in Object Oriented

Environment" International Journal of Computer Science & Engineering Technology (IJCSET)

[22] Mustafa Ghanem Saeed, Kamaran Hama Ali Faraj " Three Levels Quality Analysis Tool for Object Oriented Programming" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 11, 2018

[23] Mohammad Ibraigheeth (Abu-Ayyash), Syed Abdullah Fadzli "Software Reliability Prediction In Various Software Development Stages" Journal of Theoretical and Applied Information Technology 15th April 2018. Vol.96. No 7 ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[24] Yeresime Suresh, JayadeepPati, Santanu Ku Rath 2nd International Conference on Communication, Computing & Security [ICCCS-2012] www.sciencedirect.com Procedia Technology 6 (2012) 420 – 427.

[25] MuktamyeeSarker "An overview of Object Oriented Design Metrics"

[26] Sheena Nanda "Evaluation of Feature Selection Technique for Software Maintenance Prediction"

[27] GurpreetKour "Validating Software Evolution of Agile Projects Using Lehman Laws"

[28] Dalila et al "Towards a New Framework of Software Reliability Measurement Based on Software Metrics" The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017) www.sciencedirect.com

[29] Nidhi Gupta1, Dr. Rahul Kumar "Reliability Measurement of Object Oriented Design: Complexity Perspective" International Advanced Research Journal in Science, Engineering and Technology Vol. 2, Issue 4, April 2015. ISSN (Online) 2393-8021 ISSN 2394-1588

[30] Le Hoang Son, et al "Empirical Study of Software Defect Prediction:A Systematic Mapping" Symmetry MDPI

[31] http://madeyski.e-informatyka.pl/tools/software-defect-prediction/

[32] Yeresime et al Statistical and Machine Learning Methods for Software Fault Prediction Using CK Metric Suite: A Comparative Analysis" Hindawi Publishing Corporation ISRN Soware Engineering Volume 2014, Article ID 1083, 15 pages http://dx.doi.org/10.1155/2014/2510s