I.J. Intelligent Systems and Applications, **2019, 2, 36-48**
Published Online February 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2019.02.05

Optimal Machine learning Model for Software Defect Prediction

Tripti Lamba
Research Scholar, JaganNathUniversity, Jaipur, India
E-mail: triptigautam@yahoo.co.in

Kavita and A.K.Mishra
Associate Professor, JaganNathUniversity, Jaipur , India and Principal Scientist AKMU, IARI Pusa Campus, New Delhi, India
E-mail: kavita.yogen@gmail.com and akmishra.usi.iari@gmail.com

Received: 06 November 2017; Accepted: 12 July 2018; Published: 08 February 2019

Abstract—Machine Learning is a division of Artificial Intelligence which builds a system that learns from the data. Machine learning has the capability of taking the raw data from the repository which can do the computation and can predict the software bug. It is always desirable to detect the software bug at the earliest so that time and cost can be reduced. Feature selection technique wrapper and filter method is used to find the most optimal software metrics. The main aim of the paper is to find the best model for the software bug prediction. In this paper machine learning techniques linear Regression, Random Forest, Neural Network, Support Vector Machine, Decision Tree, Decision Stump are used and comparative analysis has been done using performance parameters such as correlation, R-squared, mean square error, accuracy for software modules named as ant, ivy, tomcat, berek, camel, lucene, poi, synapse and velocity. Support vector machine outperform as compare to other machine learning model.

Index Terms—Linear Regression, Random Forest, Neural Network, Support Vector Machine, Decision Tree, Decision Stump.

I. Introduction

Today's era is not only a computer age but an age of Artificial Intelligence wherein machine will think by itself. Machine learning technique is an important branch of computers for software bug prediction. Software bug is one of the major issues in a computer industry. It is always desirable to have minimum software bug and software system to reach at the maximum accuracy level. In this paper regression method [3],[10] is applied on machine learning model namely Linear Regression, Random Forest, Neural Network, Support Vector Machine [2], Decision Tree and Decision Stump. The objective of this paper is to identify the finest machine learning model used for the software bug prediction.

In this paper dataset was taken from open science promise repository which is publicly available at http://openscience.us/repo/defect/ck/ and the machine learning techniques were applied on software modules named as ant, ivy, tomcat, berek, camel, lucene, poi, synapse and velocity to find the best machine learning model, for this two sets of experiment was conducted one using all the variables and another using only important variable that was derived from feature selection method. Different performance comparison parameter like correlation, R Squared, mean square error (MSE) and accuracy was calculated on different machine learning models. Remaining section of the paper was arranged as follows. Section II describes the related work. Section III describes the different machine learning model used for regression. Research methodology used in the paper is described in section IV. Experiment was conducted in section V to find the best model with respect to the performance parameter used in machine learning technique. Section VI derives the result and the conclusion is drawn in section VII.

II. Related Work

Suresh *et.al.* [5] describe that software Metrics are beneficial to evaluate the complexity, software reliability and fault proneness of the system. CK metrics are the best indicators for fault proneness. Regression analyses are utilized to determine the relationship between the values of the metrics and bugs associated in the class. The DIT and NOC cannot be used for fault detection. LOC, LCOM and WMC are the best indicators for the system reliability. Singh and Salarai [29] in their study has collected data from the PROMISE repository of empirical software engineering data. This dataset uses the CK (Chidamber and Kemerer) OO (object-oriented) metrics. The accuracy of Levenberg-Marquardt (LM) algorithm based neural network is compared with the polynomial function-based neural network predictors for detection of software defects. Their results indicate that the prediction model has a high accuracy. Okutan *et.al.* [30] used Bayesian network to check the influence among software metrics and defect proneness. In combination with the metrics used in Promise Repository; they have defined two more metrics, Source Code Quality Metrics (LOCQ) and a Number Of Developers (NOD). For their experiment, they have selected nine datasets from the Promise Repository. They derived that RFC, LOC and LOCQ are more effective to defect proneness whereas NOC and DIT are less effective and unreliable. Their future work will include other software metrics and process metrics to determine the best metrics used for defect prediction and emphasis has been given to work with a smaller set of software metrics. A.Kaur et.al [31] in their research article has depicted that there are many approaches for predicting bugs in software systems. Their paper uses the metrics derived using entropy of changes to compare five machine learning techniques, namely Gene Expression Programming (GEP), General Regression Neural Network, Locally Weighted Regression, Support Vector Regression (SVR) and Least Median Square Regression for predicting bugs.The data extraction for the validation purpose is automated by developing an algorithm that employs web scraping and regular expressions. The study suggests GEP and SVR as stable regression techniques for bug prediction using entropy of changes. Rong et.al [32] in their article has pointed out the value of parameters of SVM model has a remarkable influence on its classification accuracy and the selection process lacks theory guidance that makes the SVM model uncertainty and low efficiency. In their paper, a CBA-SVM software defect prediction model is proposed, which take advantage of the non- linear computing ability of SVM model and optimization capacity of bat algorithm with centroid strategy (CBA). Through the experimental comparison with other models, CBA-SVM is proved to have a higher accuracy.

III. Machine Learning Models

Machine learning is a branch of Artificial Intelligence which will build a system that learns from and make prediction on data. Machine learning has become one of the hot topics as everyone wants to build an intelligent application [22]. Machine learning can be classified as unsupervised learning and supervised learning [20]. Unsupervised learning is a method to find the hidden patterns in input data. Clustering is unsupervised learning technique. Supervised learning [1] is used when need to train the model to make a prediction. Supervised learning can be categorized into two types: regression and classification. This paper applied regression technique on machine learning models to predict the best model for the software bug prediction. This paper used six machine learning models Linear Regression, Random Forest, Decision Tree, Support Vector Machine, Neural Network and Decision Stump.

A. Linear Regression

Linear Regression is generally used for the predictive analysis. This model finds the relationship between the response variable (dependent variable) and one or more explanatory variable (independent variable).

B. Random Forest

Random forest[23] is one of the ensemble learning user friendly methods[11] used in prediction for better performance and can be used for the software bug prediction. It is also used to rank the importance of the variable. In Random forest two parameters are used and each node is split and randomly chosen to find the best predictor.

C. Decision Tree

Decision tree is one of the supervised learning methods used in classification and regression for predictive modeling approach [21]. Decision tree has the capability to handle datasets which have errors and missing values [12]. But one of the drawbacks of decision tree is oversensitivity to the training set which is not relevant or noisy data.

D. Support Vector Machine

Support Vector Machine is supervised machine learning used in classification, regression [14] and outliers detection. SVM works well when data sets are small because the required training time is less. If the data set are less noisy, it provides a good model [24].SVM is used in many applications like face recognition field [15], Optical character Recognition [16], spam categorization [17], financial time series forecasting [18] etc.

E. Neural Network

Neural Network can be used to find the correlation between input and output, to predict software defects and find the pattern.

F. Decision Stump

Decision Stump is a machine learning model. Decision Stump can be called as one level decision tree [14]. Decision stump generally gives best result or continues to improve when feature selected has useful values.

IV. Research Methodology

Research methodology is also one of the important components to achieve the goal of any system. The main objective of the paper is to find the best machine learning model for software bug prediction. Systematic structuring is required to accomplish this objective and the framework for software defect prediction using historical datasets as shown in Fig 1.

A. Data Collection

Data collection is also one of the vital sections of a system to work upon. For the experiment/analysis dataset is collected from the open source Promise repository which is authentic and publically available. Different software metrics like product metrics, process metrics are available for software bug prediction. Chidamber & Kemerer object-oriented (CK_OO) [5] metrics suite which is a product metrics were taken for experiment.

Software modules used in the paper is shown in Table 1. Six CK metrics are Weighted Methods per Class (WMC),Response for a Class (RFC), Coupling between objects (CBO), Lack of Cohesion (LCOM), Number of children (NOC), Depth Of Inheritance (DIT) and OO metrics are Afferent couplings(CA), Efferent couplings (CE), Number of Public Methods (NPM), Lack of firmness in methods (LCOM3), Lines of Code (LOC), Data Access Metric (DAM), Measure of aggregation (MOA), Measure of Functional Abstraction (MFA), Cohesion Among Methods of Class (CAM), Inheritance Coupling (IC), Coupling Between Methods (CBM), Average Method Complexity (AMC) and McCabe's cyclomatic complexity (CC - MAX_CC, AVG_CC).

.

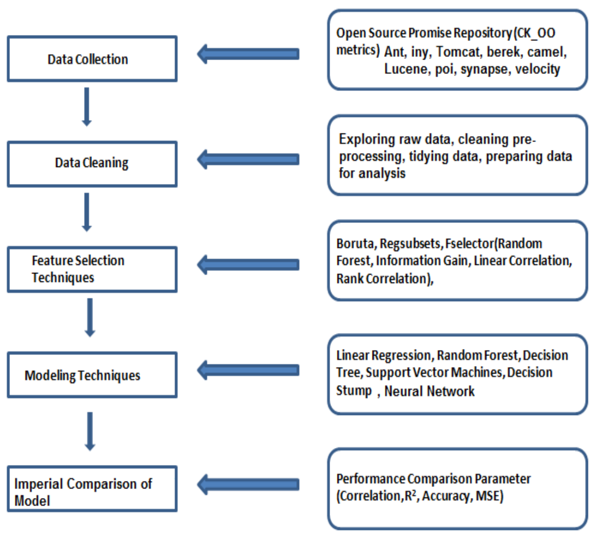

Data Collection
Open Source Promise Repository(CK_OO metrics) Ant, iny, Tomcat, berek, camel, Lucene, poi, synapse, velocity
Data Cleaning
Exploring raw data, cleaning pre-processing, tidying data, preparing data for analysis
Feature Selection Techniques
Boruta, Regsubsets, Fselector(Random Forest, Information Gain, Linear Correlation, Rank Correlation),
Modeling Techniques
Linear Regression, Random Forest, Decision Tree, Support Vector Machines, Decision Stump , Neural Network
Imperial Comparison of Model
Performance Comparison Parameter (Correlation,R², Accuracy, MSE)

Fig.1. Framework for Software Defect prediction using Historical Databases

B. Data Cleaning

Data Cleaning [19] is helpful in improving the quality of the data as it deals in detecting inconsistencies, removing errors, missing values [22]. Rahm et.al [24] in their paper addresses the issue of data cleaning which is a foremost part of extraction, transformation, loading (ETL) process in a data warehouse. Various types of tools [25] are available to clean the data but sometime major portion of the data need to be clean manually that are difficult to write and maintain.

C. Feature Selection

Feature Selection is an extremely significant phase in bug prediction for accuracy and the complication of the model [6]. Another vital factor of using the Feature Selection is that if the number of variables is higher than optimal, then the machine learning algorithm exhibits a decrease in accuracy. So it is desirable that selection of small feature set gives best result. Various types of Feature Selection techniques, Boruta [8], Regsubset [26], FSelector [13] can be utilized to explore the optimal metrics for software bug prediction. The Feature Selection uses Wrapper, Filter and Hybrid Algorithm.

D. Mathematical Model

One of the machines learning technique is a regression [1] which can be used to formulate the prediction model [2]. This paper has applied performance parameters [4] Correlation, R-squared, Mean Square Error and Accuracy in machine learning models. Performance parameters were calculated as follows:

- Correlation (corr) [4]

Correlation can be defined as the association between actual and predicted values. The values lie between 0 and 1 and the value of correlation which is near to 1 is considered as good. Mathematical representation is as follows:

$$corr = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}} \quad (1)$$

Where, x = Actual Value; y = Predicted Value; $\bar{x}$ =mean of all the actual values; $\bar{y}$ =mean of all the Predicted values; n = total number of instances

- R-Squared (R^2)

R Squared is known as coefficient of determination. The coefficient of determination R-Squared is used to analyze how differences in one variable can be explained by a difference in a second variable and determine as percent. The higher the coefficient, the higher percentage of points the line passes through when the data points and line are-plotted. Values of 1 or 0 would indicate the regression line represents all or none of the data respectively.

$$R^2 = corr * corr \quad (2)$$

- Mean Square Error(MSE)

Mean Square Error is used to calculate the error rate of a regression model.

$$MSE = \frac{\sum_{i=1}^{n} (p_i - a_i)^2}{n} \quad (3)$$

Where, a = actual target; p = predicted target; n = total number of instances

- Accuracy

The accuracy is calculated as percentage deviation of predicted target with actual target with acceptable error.

$$\frac{100}{n} \sum_{i=1}^{n} q_i$$
$$q_i = \{^{1}_{0} \quad if\, abs(p_i - a_i) \leq err \quad (4)$$

Where, a = actual target; p = predicted target; n = total number of instances; err=acceptable error.

Table 1. Software Modules

Software Modules	Versions	No. of Observations	No. Of variables
Ant	1.7	745	21
Ivy	2.0	352	21
Tomcat	6.0	858	21
Berek	1.0	44	21
Camel	1.6	965	21
Lucene	2.4	340	21
Poi	3.0	442	21
Synapse	1.2	256	21
Velocity	1.6	229	21

V. Experiment

For the experiment the data set is split as 50% training data set and 50% testing data set, which validates the values of statistical measures reported by D'Ambros et.al (2010) for CK and OO metrics[5]. These have been implemented and simulated in RStudio (R 3.3.1) environment. Feature selection [6] is a very important phase in the bug prediction for the accuracy and the complication of the model [7]. Another important factor of using feature selection is that if the numbers of dependent variables are more than optimal than machine learning algorithm show a decrease of accuracy. So it is desirable that selection of small feature set gives best result. Feature selection uses wrapper, filter and hybrid algorithm. Different types of feature selection technique Boruta [8], Regsubset [9], FSelector [13] are used to get the most optimal metrics for software bug prediction. The Filter algorithm FSelector like Random Forest, Information Gain, Rank Correlation and Linear Correlation are used in this paper.

A. *Boruta Feature Selection Technique*

Boruta is a Wrapper Algorithm, which is used to discover the relevant variables. It performs iteration and removes the variables, which are less relevant. When Boruta is run, Z-Scores boxplot graph is drawn as shown in Fig. 2, the green boxplot represent a Z-Scores of confirmed (important) variables, blue boxplot represents a Z-Scores of average variable and red boxplot represents a Z-Scores of rejected variables for the Ant software modules. The variable (software metrics) accepted by running the Boruta Algorithm for Ant Software modules are MOA, MFA, CAM, AMC, NPM, LCOM, CE, AVG_CC, WMC, MAX_CC, LOC and RFC. The variables rejected are IC, NOC, CBM, CA, DIT and DAM. The variable which can be accepted or rejected that is average is LCOM3 and CBO. For other software modules the Z-Scores are calculated and the software metrics accepted and rejected by the different software modules using Boruta are represented as 'T' and 'F' respectively as shown in Table 2 and the reduce set of variable for Boruta and Regsubset is displayed in Table 3.

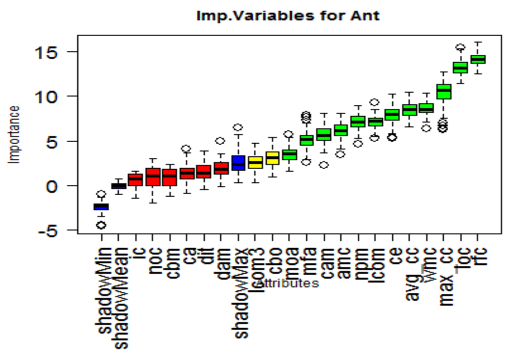


Fig.2. Feature Selection using Boruta for Ant Software Modules

1. *Significant Software Metrics Assortment Using Boruta*

To find the significance of the software metrics the total number of "T" was calculated from Table 2 for each software metrics such as WMC, DIT, NOC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM,IC, CBM, AMC, MAX_CC, and AVG_CC for all the nine software modules shown in fig 3.

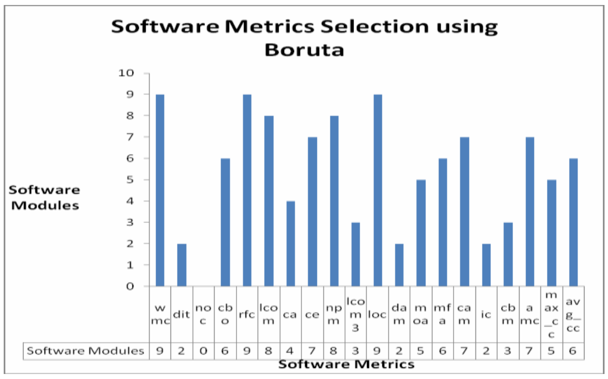


Fig.3. Software Metrics Selection using Boruta for the Software Modules Ant, Ivy, Tomcat, Berek, Camel, Lucene, Poi, Synapse and Velocity

B. *Regsubset Feature Selection Technique*

The other feature selection technique used in this paper is Regsubset. It uses exhaustive selection algorithm and subset of size eight. The number is been allotted to each of the metrics out of eight for different software modules shown in Table 4. Two indicators black and white are shown in fig 4, black indicates that variables are in the model or accepted software metrics which includes WMC, RFC, LCOM, CE, NPM, LOC, MOA, AMC whereas white indicate variables are not in the model and they are DIT, NOC, CBO, CA, LCOM3, DAM, MFA, CAM, IC, MAX_CC, AVG_CC and CBM. Similarly for the remaining software modules Regsubset feature selection technique. The reduce set of variable for Boruta and Regsubset is displayed in Table 3.

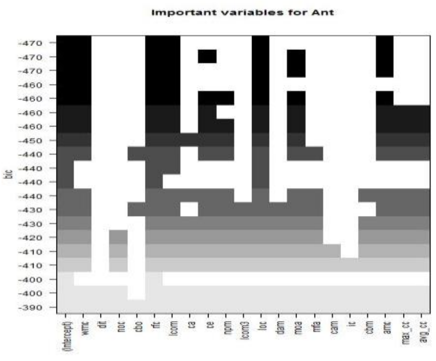
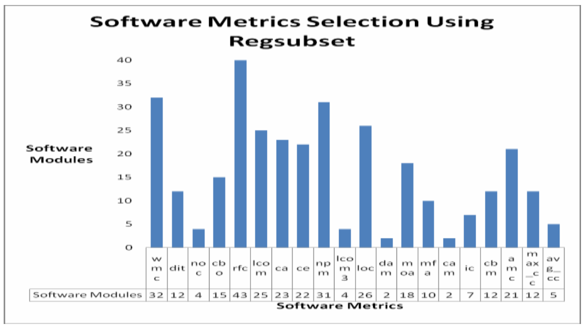


Fig.4. Feature Selection using Regsubset for Ant Software Module

1. Significant Software Metrics Assortment Using Regsubset

The significance of the software metrics in a Feature Selection technique, regsubset is computed by adding each of the software metrics resp. for all the nine software modules from Table 4 and plotted in Fig 5. The most significant metrics are RFC value as 43, WMC as 32 and LOC as 26.The least significant metrics are CAM and DAM as 0 then NOC as 4 is shown in fig 5.



	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	max_cc	avg_cc
Software Modules	32	12	4	15	43	25	23	22	31	4	26	2	18	10	2	7	12	21	12	5

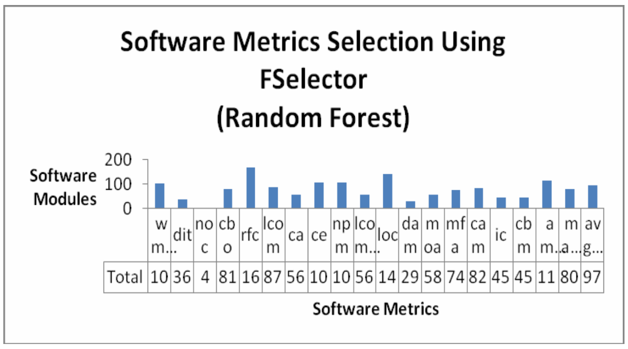
Fig.5. Software Metrics Selection using Regsubset for the Software Modules Ant,Ivy,Tomcat, Berek,Camel,Lucene,Poi,Synapse and Velocity

C. FSelector Feature Selection Technique

FSelector is another feature selection technique used in this paper. The correlation filter such as linear correlation and rank correlation uses Pearson’s correlation and Spearman’s correlation respectively are used. This algorithm finds weights of continuous attributes based on their correlation with continuous class attribute package FSelector. Table 5, 6, 7 and 8 shows weighted attribute using Random Forest, Information Gain, Linear correlation and Rank Correlation respectively.

1. Significant Software Metrics Assortment Using FSelector (Random Forest)

The graph plotted in Fig. 6 using FSelector (Random Forest) indicate that the most significant software metrics are RFC having a value 168, LOC value as 140, AMC value as 113, NPM value as 106 and so on. The least significant software metrics are NOC value as 4, then DAM as 29 and DIT as 36.

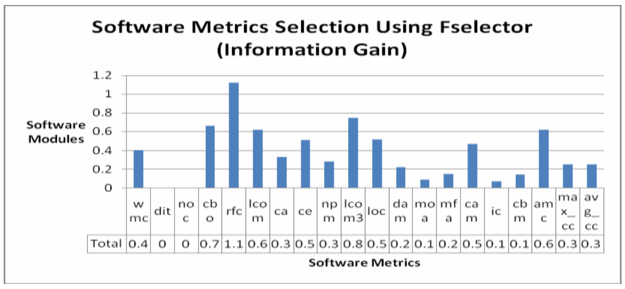


	wm...	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom...	loc	dam	moa	mfa	cam	ic	cbm	am...	ma...	avg...
Total	10	36	4	81	16	87	56	10	10	56	14	29	58	74	82	45	45	11	80	97

Fig.6. Software Metrics Selection Using FSelector(Random Forest) for the Software Modules Ant,Ivy, Tomcat, Berek, Camel, Lucene, Poi, Synapse, Velocity

2. Significant Software Metrics Assortment Using FSelector (Information Gain)

The graph plotted using FSelector Information Gain) in Fig 7 indicate that most significant software metrics are RFC having a value 1.1, LCOM3 value as 0.8, AMC and LCOM value as 0.6. The least significant software metrics are NOC and DIT as 0.

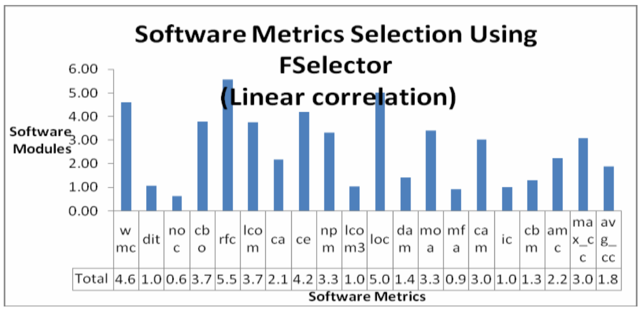


	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	max_cc	avg_cc
Total	0.4	0	0	0.7	1.1	0.6	0.3	0.5	0.3	0.8	0.5	0.2	0.1	0.2	0.5	0.1	0.1	0.6	0.3	0.3

Fig.7. Software Metrics Selection Using FSelector(Information Gain) for the Software Modules Ant,Ivy, Tomcat, Berek, Camel, Lucene, Poi, Synapse, Velocity

3. Significant Software Metrics Assortment Using FSelector (Linear Correlation)

The graph plotted in Fig 8 using FSelector (Linear Correlation) indicates that the most significant software metrics are RFC having a value 5.5, LOC as 5 and WMC as 4.6. The least significant software metrics are NOC value as 0.6, DIT and IC as 1.0.



	wmc	dit	noc	cbo	rfc	lcom	ca	ce	npm	lcom3	loc	dam	moa	mfa	cam	ic	cbm	amc	max_cc	avg_cc
Total	4.6	1.0	0.6	3.7	5.5	3.7	2.1	4.2	3.3	1.0	5.0	1.4	3.3	0.9	3.0	1.0	1.3	2.2	3.0	1.8

Fig.8. Software Metrics Selection Using FSelector (Linear correlation) for the Software Modules Ant, Ivy, Tomcat, Berek, Camel, Lucene,synapse and velocity.

4. Significant Software Metrics Assortment Using FSElector (Rank correlation)

The graph plotted using FSelector (Rank Correlation) in Fig 9 indicates that the most significant software metrics are RFC having a value 4, LOC as 3.9, WMC as 3.7. The least significant software metrics are DIT as 0.6, NOC and MFA as 0.8.

D. Significant and Insignificant metrics

The significance of various software metrics using Feature Selection techniques is plotted in Figures 3, 5, 6, The graph plotted using FSelector (Rank Correlation) in Fig 9 indicates that the most significant software metrics are RFC having a value 4, LOC as 3.9, WMC as 3.7. The insignificant software metrics selected using Feature Selection technique for software modules: Ant, Ivy, Tomcat, Camel, Berek, Lucene, POI, Synapse and Velocity is shown in Table 9.

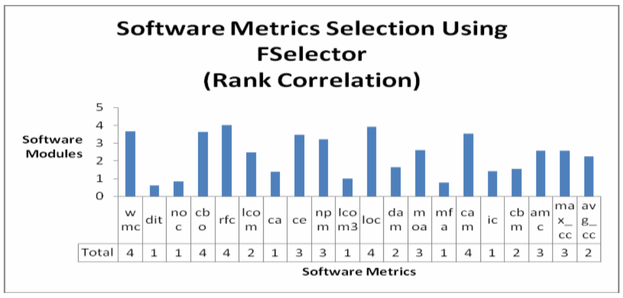


Fig.9. Software Metrics Selection Using FSelector (rank Correlation) for thr Software Modules Ant, Ivy, Tomcat, Berek, Camel, Lucene, Poi, Synapse, Velocity.

E. Snippets of machine learning algorithm used with Tuning Parameters

The R packages used in machine learning models with the tuning parameters are shown as in Table 10. These have been implemented and simulated in R Studio (R.3.3.1) environment.

Table 2. Confirmed Software Metrics using Boruta

Software modules	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	NPM	LCOM3	LOC	DAM	MOA	MFA	CAM	IC	CBM	AMC	MAX_CC	AVG_CC
Ant	T	F	F	F	T	T	F	T	T	F	T	F	T	T	T	F	F	T	T	T
Ivy	T	F	F	T	T	T	F	T	T	F	T	F	F	F	T	F	F	T	T	T
Tomcat	T	F	F	F	T	T	F	F	T	F	T	F	F	T	F	F	F	F	F	F
Berek	T	T	F	T	T	T	T	T	F	F	T	F	T	T	F	F	F	T	F	F
Camel	T	F	F	F	T	T	F	F	T	T	T	T	T	F	T	F	F	T	T	T
Lucene	T	T	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	T
Poi	T	F	F	T	T	T	T	T	T	T	T	F	T	T	T	F	T	T	T	T
Synapse	T	F	F	T	T	F	T	T	T	F	T	F	F	T	T	F	F	T	T	T
Velocity	T	F	F	T	T	T	F	T	T	F	T	F	F	F	T	T	T	T	F	F

Table 3. Reduced set of CK and OO Software Metrics using Boruta and Regsubset

Software Modules	No. Of Variables	Feature Selection using Boruta		Feature Selection Using Regsubset	
		Reduced Variables	Reduced Software metrics	Reduced Variables	Reduced Software metrics
Ant	21	12	WMC, RFC, LCOM, CE, NPM, LOC, MOA, MFA, CAM, AMC,MAX_CC, AVG_CC	8	WMC, RFC, LCOM, CE, NPM, LOC, MOA, AMC
Ivy	21	11	WMC, CBO, RFC, LCOM, CE, NPM, LOC, CAM, AMC, MAX_CC, AVG_CC	9	WMC, NOC, LCOM, CE, NPM, LOC, MFA, MAX_CC, AVG_CC
Tomcat	21	6	WMC, RFC, LCOM, NPM, LOC, MFA	9	WMC, CBO, RFC, CA, NPM,LCOM3, DAM, MOA, CBM
Berek	21	11	WMC, DIT, CBO, RFC, LCOM, CA, CE, LOC, MOA, MFA, AMC	14	WMC, DIT, CBO, RFC, LCOM, CA, CE,NPM, LOC, MOA, MFA, CBM, AMC, MAX_CC
Camel	21	12	WMC, RFC, LCOM, NPM, LCOM3, LOC, DAM, MOA, CAM, AMC, MAX_CC, AVG_CC	9	WMC, CBO, LCOM, CA, CE, NPM, LOC,DAM, MOA
Lucene	21	17	WMC, DIT, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, DAM, MOA, MFA, CAM, IC, CBM, AVG_CC	10	WMC, NOC, CBO, RFC, LCOM, CA, CE,NPM, IC, CBM
Poi	21	16	WMC, CBO, RFC, LCOM, CA, CE, NPM, LCOM3, LOC, MOA, MFA, CAM, CBM, AMC, MAX_CC, AVG_CC	7	WMC, DIT, RFC, LCOM, CE, MFA, CBM
Synapse	21	12	WMC, CBO, RFC, LCOM, CE, NPM, LOC, CAM, IC, CBM, AMC	9	WMC, DIT, CBO, RFC, CA, NPM, LOC, MFA, AMC
Velocity	21	11	WMC, CBO, RFC, LCOM, CE, NPM, LOC, CAM, IC, CBM, AMC	9	WMC, CE, LOC, MOA, CAM, IC, AMC,MAX_CC, AVG_CC

Table 4. Important Metrics using Regsubset

Software modules	WMC	DIT	NOC	CBO	RFC	LCOM	CA	CE	NPM	LCOM3	LOC	DAM	MOA	MFA	CAM	IC	CBM	AMC	MAX_ CC	AVG _CC
Ant	4	0	0	0	8	6	0	2	1	0	7	0	3	0	0	0	0	4	0	0
Ivy	6	0	3	0	0	4	0	6	5	0	2	0	0	1	0	0	0	0	3	3
Tomcat	4	0	0	3	8	0	6	0	7	4	0	1	5	0	0	0	2	0	0	0
Berek	1	7	0	2	3	4	2	1	3	0	5	0	1	4	0	0	1	4	7	0
Camel	5	0	0	5	0	3	7	3	5	0	1	1	8	0	0	0	0	0	0	0
Lucene	3	0	1	2	8	5	6	1	3	0	0	0	0	0	0	2	5	0	0	0
POI	4	3	0	0	8	3	0	4	0	0	0	0	0	3	0	0	4	0	0	0
Synapse	2	2	0	3	8	0	2	0	7	0	4	0	0	2	0	0	0	6	0	0
Velocity	3	0	0	0	0	0	0	5	0	0	7	0	1	0	2	5	0	7	2	2

Table 5. Feature Selection Using FSelector (Random Forest)

	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Poi	Synapse	Velocity
WMC	13.5	11	6.9	7.2	9.9	12.6	10.2	5.7	12
DIT	1.3	1.6	3.9	6.4	6.3	6.2	3	2.7	2.3
NOC	3.9	-1.1	-1.6	0	1.8	1.2	2.3	-6	2.1
CBO	7.2	9.3	8.2	7	2.3	15	9.2	7	8.5
RFC	22.3	13.6	17.2	11.7	6.9	20.6	20.6	19.1	11.8
LCOM	10.7	10.8	3.4	7	9.3	13.2	8.8	4.8	8.7
CA	3.4	6.5	2.8	7.1	5.6	11.6	6.2	4.7	4.3
CE	12.4	13.5	0	8.7	2.1	16.9	13.5	10.9	17.6
NPM	13.4	16.9	13.4	5.6	7	14.7	10.5	3.8	9.1
LCOM3	7	3	8.4	2.1	9.5	5.9	6	3.8	3.8
LOC	21.8	16.2	12.7	10.6	7	6.6	12.5	15.2	17.9
DAM	3.2	0.8	1.1	3.5	5.5	9.4	4.5	0	-1.4
MOA	7.4	1.4	2	6.1	17.4	2.6	6.2	4.6	3
MFA	11.6	2.4	10.7	4.2	7	9.2	3.4	8.5	5.9
CAM	11.6	9.5	-2.4	1.1	10	8.6	10.7	7.5	13.9
IC	1.4	1.2	0.8	-0.2	2.6	18.8	4	4.7	10.9
CBM	1.9	2.2	-1.9	1.2	2.2	12.7	10.3	4.6	9.4
AMC	11.1	10	4.9	13.6	11.9	3.8	13.4	16.1	16.3
MAX_CC	18.1	3.3	1.2	3.3	6.1	3.9	8.9	13.1	2.2
AVG_CC	19	6.3	2.6	2.4	9.2	7.5	11.6	13.5	5.4

Table 6. Feature Selection using FSelector (Information Gain)

	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Poi	Synapse	Velocity
WMC	0.1	0	0	0	0	0.1	0.1	0	0
DIT	0	0	0	0	0	0	0	0	0
NOC	0	0	0	0	0	0	0	0	0
CBO	0.1	0	0	0.3	0	0.1	0.1	0	0
RFC	0.1	0.1	0	0.4	0	0.1	0.2	0.1	0
LCOM	0.1	0	0	0.3	0	0.1	0.1	0	0
CA	0	0	0	0.3	0	0	0	0	0
CE	0.1	0	0	0	0	0.1	0.2	0.1	0.1
NPM	0.1	0	0	0	0	0.1	0.1	0	0
LCOM3	0.1	0	0	0.4	0	0	0.2	0	0
LOC	0.1	0.1	0	0	0	0	0.1	0.1	0
DAM	0	0	0	0	0	0.1	0.1	0	0
MOA	0	0	0	0	0	0	0	0	0
MFA	0	0	0	0	0	0	0.1	0	0
CAM	0.1	0.1	0	0	0	0.1	0.1	0	0.1
IC	0	0	0	0	0	0	0.1	0	0
CBM	0	0	0	0	0	0	0.1	0	0
AMC	0.1	0	0	0.3	0	0	0.1	0.1	0
MAC_CC	0.1	0	0	0	0	0	0.1	0	0
AVG_CC	0	0	0	0	0	0	0.2	0	0

Table 7. Feature Selection using FSelector (Linear Correlation)

	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Poi	Synapse	Velocity
WMC	0.43	0.41	0.3	0.63	0.31	0.7	0.53	0.27	0.48
DIT	0.05	0.02	0.02	0.58	0.01	0.06	0.13	0.05	0.1
NOC	0.1	0.02	0.06	0.08	0.15	0.1	0.03	0.07	0.01
CBO	0.35	0.3	0.3	0.73	0.46	0.48	0.31	0.32	0.26
RFC	0.49	0.51	0.45	0.75	0.28	0.73	0.69	0.49	0.53
LCOM	0.41	0.22	0.18	0.68	0.14	0.65	0.38	0.15	0.43
CA	0.12	0.07	0.12	0.73	0.42	0.28	0.12	0.08	0.06
CE	0.37	0.46	NA	0.75	0.2	0.55	0.44	0.44	0.47
NPM	0.37	0.31	0.2	0.11	0.27	0.64	0.46	0.19	0.27
LCOM3	0.02	0.07	0.08	0.14	0.05	0.13	0.14	0.17	0.13
LOC	0.49	0.52	0.43	0.74	0.28	0.57	0.37	0.44	0.53
DAM	0.15	0.15	0.18	0.03	0.05	0.22	0.17	0.19	0.11
MOA	0.34	0.32	0.31	0.66	0.24	0.32	0.24	0.22	0.41
MFA	0.07	0.07	0.03	0.36	0.06	0.04	0.13	0.06	0.07
CAM	0.4	0.28	0.22	0.4	0.19	0.33	0.26	0.27	0.33
IC	0.13	0.05	0.11	0.2	0	0.23	0.04	0.02	0.1
CBM	0.13	0.11	0.18	0.2	0.04	0.35	0.06	0.04	0.09
AMC	0.35	0.18	0.13	0.67	0.06	0.07	0.09	0.16	0.47
MAX_CC	0.38	0.4	0.3	0.32	0.21	0.11	0.35	0.25	0.32
AVG_CC	0.31	0.2	0.23	0.27	0.16	0.01	0.2	0.09	0.14

Table 8. Feature Selection using FSelector (Rank Correlation)

	Ant	Ivy	Tomcat	Berek	Camel	Lucene	Poi	Synapse	Velocity
WMC	0.43	0.34	0.31	0.31	0.31	0.42	0.48	0.33	0.31
DIT	0.05	0.03	0.11	0.11	0.01	0.12	0.01	0.03	0.11
NOC	0.1	0.03	0.1	0.1	0.15	0.09	0.02	0.03	0.1
CBO	0.35	0.3	0.31	0.31	0.46	0.41	0.47	0.32	0.31
RFC	0.49	0.36	0.33	0.33	0.28	0.45	0.54	0.43	0.33
LCOM	0.41	0.3	0.15	0.15	0.14	0.19	0.37	0.21	0.15
CA	0.12	0.07	0.02	0.02	0.42	0.18	0.21	0.24	0.02
CE	0.37	0.2	0.4	0.4	0.2	0.38	0.49	0.28	0.4
NPM	0.37	0.27	0.3	0.3	0.27	0.43	0.4	0.22	0.3
LCOM3	0.02	0.01	0.14	0.14	0.05	0.14	0.19	0.16	0.14
LOC	0.49	0.36	0.34	0.34	0.28	0.35	0.5	0.44	0.34
DAM	0.15	0.13	0.12	0.12	0.05	0.32	0.28	0.2	0.12
MOA	0.34	0.18	0.29	0.29	0.24	0.13	0.25	0.29	0.29
MFA	0.07	0.06	0.14	0.14	0.06	0.06	0.02	0.01	0.14
CAM	0.4	0.3	0.4	0.4	0.19	0.36	0.37	0.32	0.4
IC	0.13	0.04	0.15	0.15	0	0.26	0.29	0.11	0.15
CBM	0.13	0.05	0.16	0.16	0.04	0.25	0.35	0.12	0.16
AMC	0.35	0.27	0.24	0.24	0.06	0.17	0.36	0.28	0.24
MAX_CC	0.38	0.29	0.1	0.1	0.21	0.24	0.47	0.32	0.1
AVG_CC	0.31	0.26	0.14	0.14	0.16	0.16	0.41	0.23	0.14

Table 9. Significant and Insignificant Metrics Using Feature Selection Techniques

Feature Selection Technique	Significant Metrics	Insignificant Metrics
Boruta	WMC,RFC,LOC, LCOM, NPM	NOC,DIT, DAM, IC
Regsubset	WMC,RFC, LOC, NPM, LCOM	DAM, NOC, CAM, NOC, LCOM3
FSelector (Random Forest)	WMC,RFC,NPM, LOC, AMC	NOC,DIT, DAM, IC, CBM
FSelector (Information Gain)	RFC, LCOM3, CBO, LCOM, AMC	NOC,DIT, MOA, IC, CBM
FSelector (Linear Correlation)	RFC, LOC, WMC, CE, CBO	DIT, NOC, LCOM3, MFA, IC
FSelector (Rank Correlation)	RFC,LOC, WMC, CBO, CE	DIT,NOC, MFA, LCOM3

F. Performance Parameters Applied

Performance parameters [4] Correlation, R-Squared, Mean Square Error and Accuracy were calculated on machine learning models.

1. Machine learning models applied combining all the Software Metrics.

The machine learning models with the tuning parameters as discussed in Table 10 was applied with all the software metrics to obtain the correlation, R-Squared, Mean Square Error and Accuracy shown in Tables 11, 12, 13 and 14 below respectively. The graph is plotted to compare the performance parameters by computing the mean average of each nine software modules with respective Machine Learning Models as shown in Figure 10, 11, 12 and 13 to get the best machine learning model.

The Correlation Comparative Analysis was done of the machine learning model using all the software metrics. Fig: 10 depicted that Random Forest has the highest correlation at 0.44 and Neural Network and Decision Stump have lowest correlation at 0.29 using all the software metrics.

The R-Squared Comparative Analysis was done of the machine learning model using all the software metrics. The Fig. 11 depicted that the Random Forest has highest R-Squared value as 0.33 and Neural Network has the lowest R Squared values as 0.18.

Support Vector Machine has the lowest mean square value as 0.74 and the Neural Network has the highest Mean Square Error as 1.7 shown in Fig: 12.

Table 10. Machine Learning Models, Packages and Tuning Parameters

ML Models	**Technique**	**Packages**	**Tuning Parameters**
Linear Regression	Lm	None	None
Random Forest	Random Forest	Library(Random Forest)	ntree=250, mtry=3
Decision Tree	Rpart	Library(rpart)	parms=list(split="information"), control =rpart.control(usesurrogate=0, maxsurrogate=0))
Support Vector machine	Svm	library(e1071)	nu=0.5, epsilon=0.1
Neural Network	Neuralnet	Library(neuralnet)	hidden = 1, threshold = 0.01,stepmax = 1e+05, rep = 1
Decision Stump	DecisionStump	library(RWeka)	control = Weka_control(), options = NULL

Table 11. Correlation Calculated combining of all the Software Metrics

Machine Leaning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	0.6	0.3	0.4	0.5	0.0	0.3	0.2	0.1	0.1
Random Forest	0.6	0.5	0.4	0.7	0.1	0.4	0.4	0.2	0.2
Neural Network	0.2	0.4	0.3	0.1	0.0	0.4	0.4	0.0	0.2
Decision Tree	0.5	0.1	0.4	0.6	0.0	0.3	0.2	0.1	0.5
SVM	0.4	0.3	0.1	0.8	0.0	0.3	0.1	0.1	0.0
Decision Stump	0.4	0.5	0.2	0.7	0	0	0.2	0.1	0

Table 12. R Squared calculated combining all the software metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	0.48	0.14	0.23	0.26	0.05	0.37	0.21	0.14	0.18
Random Forest	0.41	0.31	0.18	0.52	0.16	0.48	0.42	0.27	0.25
Neural Network	0.06	0.23	0.14	0.01	0.01	0.42	0.45	0.01	0.25
Decision Tree	0.3	0.02	0.22	0.42	0.03	0.32	0.24	0.14	0.5
SVM	0.22	0.14	0.01	0.79	0.07	0.32	0.18	0.16	0.09
Decision Stump	0.23	0.45	0.3	0.77	0.35	0.32	0.18	0.16	0.09

Table 13. Mean Square Error calculated combining all the software metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	poi	synapse	velocity
Linear Model	0.23	2.28	0.23	0.23	1.54	0.05	1.99	0.88	0.62
Random Forest	0.5	0.22	0.19	1.44	0.78	1.3	0.77	0.53	0.92
Neural Network	2.28	0.27	1.98	2.11	0.96	1.55	1.85	2.45	1.89
Decision Tree	0.51	0.18	0.18	1.25	0.85	1.51	0.94	0.63	1.18
SVM	0.42	0.18	0.27	1.34	0.64	1.63	0.78	0.58	0.81
Decision Stump	0.63	0.24	0.23	1.54	0.86	1.99	0.88	0.62	1.01

Table 14. Accuracy calculated combining all the software metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	87.67	93.22	97.21	36.6	74.95	49.71	73.42	84.5	72.17
Random Forest	85.52	96.61	97.67	54.55	83.44	52.05	76.58	86.05	66.96
Neural Network	34.58	95	34.42	40.91	77.64	47.95	41.89	45.74	60
Decision Tree	87.4	97.74	96.28	68.18	85.3	56.14	80.63	81.4	62.61
SVM	90.62	97.18	97.74	72.73	88.41	55.56	81.98	88.37	74.78
Decision Stump	89.0	96.6	96.05	77.27	89.44	29.8	80.6	82.17	84.35

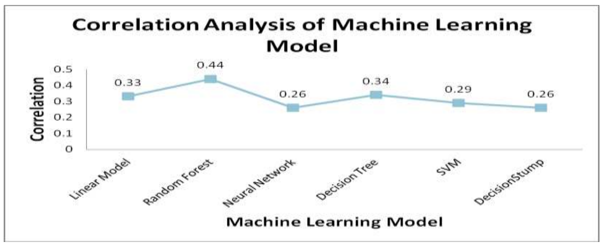


Fig.10. Correlation Analysis comparison of machine learning Model using all the software metrics

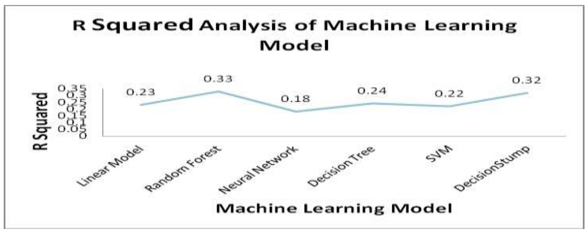


Fig.11. R-Squared Analysis Comparison of Machine Learning Model using all the Software Metrics,

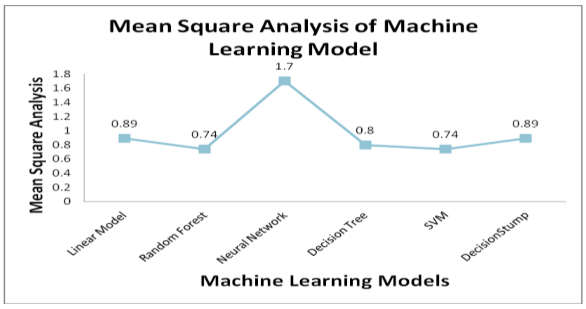


Fig.12. Mean Square Error Analysis Comparison of Machine Learning Model using all the software metrics

The Accuracy Comparative Analysis was done of the Machine Learning Model using all the software metrics. Fig.13 depicted that the Support Vector machine has the highest value as 83.04 and the Neural Network has the lowest accuracy as 53.13.

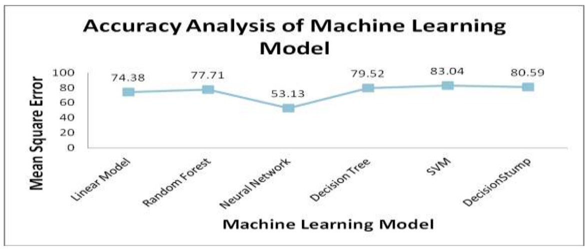


Fig.13. Accuracy Analysis Comparison of Machine Learning Model using all the software metrics

2. Machine learning models applied combining WMC, RFC and LOC Software Metrics.

The Performance Parameters Correlation, R-Squared, Mean Square Error and Accuracy as shown in Table 15, 16, 17 and 18 respectively were computed when only optimal software metrics WMC, RFC and LOC were applied on a machine learning model, Linear Regression, Random Forest, Decision Tree, Neural Network, Support Vector Machine and Decision Stump.

VI. Result

A. *Optimal Machine Learning Model*

The comparative analysis was done by using the performance parameters on machine learning models discussed in Section II on the software modules described in table 1. Two observations were analyzed which are described below:

1. Feature Selection Analysis

When the modeling technique was applied on the reduced variable, the result was either better or the same. Another important factor of using Feature Selection technique is that if the number of variables is higher than optimal, then the Machine Learning Algorithm exhibits a decrease in accuracy. The Table 9 shows the significant and insignificant software metrics. The comparative analysis was conducted to achieve the most optimal metrics by comparing the result shown in Table 9 derives that RFC, LOC and WMC are the most optimal metrics and the least significant metrics are DIT and NOC.

2. Machine Learning Models Analysis

In order to discover the best model result, a comparative analysis of machine learning models having different performance parameters using all the CK_OO software metrics is compared optimal software metrics like WMC, RFC and LOC. It was found that the Support Vector Machine is the best model and its accuracy is 83% and mean square error is 0.7 %. Correlation, R Squared, Mean Square Error and Accuracy are calculated on different machine Learning Model using all the software metrics and with WMC, RFC and LOC software metrics is shown in Fig. 14, 15, 16 and 17 respectively

Table 15. Correlation calculated combining WMC, RFC, LOC Software Metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	0.72	0.44	0.46	0.82	0.14	0.65	0.69	0.47	0.42
Random Forest	0.58	0.42	0.37	0.72	0.26	0.69	0.6	0.44	0.64
Neural Network	0.57	0.44	0.53	0.63	0.25	0.58	0.67	0.28	0.52
Decision Tree	0.56	0.33	0.42	0.58	0.22	0.54	0.52	0.37	0.36
SVM	0.51	0.58	0.5	0.68	0.19	0.5	0.42	0.45	0.43
Decision Stump	0.48	0.39	0.33	0.64	0.07	0	0.43	0.34	0

Table 16. R Squared Calculated Combining WMC, RFC, and LOC Software Metrics

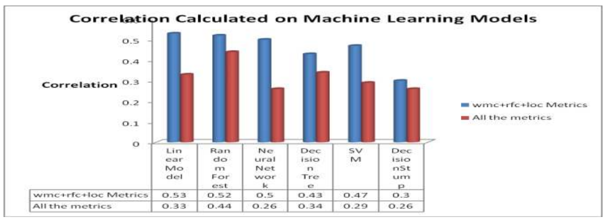
Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	0.52	0.19	0.21	0.67	0.02	0.42	0.48	0.22	0.18
Random Forest	0.34	0.18	0.14	0.52	0.07	0.48	0.36	0.18	0.41
Neural Network	0.32	0.19	0.28	0.4	0.06	0.34	0.45	0.08	0.27
Decision Tree	0.31	0.11	0.18	0.34	0.05	0.29	0.27	0.14	0.13
SVM	0.26	0.34	0.25	0.46	0.04	0.25	0.18	0.2	0.18
Decision Stump	0.23	0.15	0.11	0.41	0	0	0.18	0.12	0

Table 17. Mean Square Error Calculated Combining WMC, RFC and LOC Software Metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	0.38	0.54	0.23	0.23	1.75	0.05	1.99	0.89	0.62
Random Forest	0.5	0.21	0.19	1.41	0.84	1.57	0.81	0.56	0.89
Neural Network	0.53	0.23	0.2	1.23	0.79	1.54	0.91	0.71	1.02
Decision Tree	0.48	0.24	0.18	1.69	0.75	1.58	0.92	0.68	0.94
SVM	0.46	0.17	0.16	1.03	0.65	1.4	0.81	0.56	0.72
Decision Stump	0.63	0.23	0.23	1.75	0.86	1.99	0.89	0.62	1.01

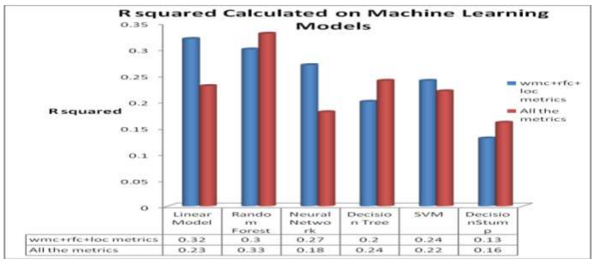
Table 18. Accuracy Calculated Combining WMC, RFC, LOC software metrics

Machine Learning Model	Ant	Ivy	Tomcat	Berek	Camel	Lucene	POI	Synapse	Velocity
Linear Model	86.33	94.92	96.05	68.18	81.57	52.63	81.08	87.6	77.3
Random Forest	86.86	93.79	97.91	68.18	80.12	47.37	77.03	82.95	65.22
Neural Network	87.4	95.48	96.98	63.64	82.82	45.61	79.28	82.95	72.17
Decision Tree	85.52	94.92	96.74	54.55	83.64	49.12	80.18	79.07	69.57
SVM	87.94	96.61	97.67	63.64	89.86	58.48	81.98	86.05	80
Decision Stump	89.01	98.31	96.28	72.73	89.44	29.82	54.05	82.17	84.35



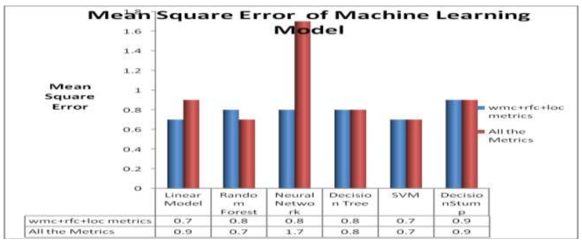
	Linear Model	Random Forest	Neural Network	Decision Tree	SVM	DecisionStump
wmc+rfc+loc Metrics	0.53	0.52	0.5	0.43	0.47	0.3
All the metrics	0.33	0.44	0.26	0.34	0.29	0.26

Fig.14. Correlation Calculated on different machine Learning Model



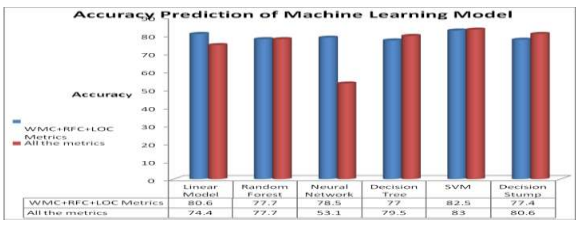
	Linear Model	Random Forest	Neural Network	Decision Tree	SVM	DecisionStump
wmc+rfc+loc metrics	0.32	0.3	0.27	0.2	0.24	0.13
All the metrics	0.23	0.33	0.18	0.24	0.22	0.16

Fig.15. R-Squared Calculated on different Machine Learning Models



	Linear Model	Random Forest	Neural Network	Decision Tree	SVM	DecisionStump
wmc+rfc+loc metrics	0.7	0.8	0.8	0.8	0.7	0.9
All the Metrics	0.9	0.7	1.7	0.8	0.7	0.9

Fig.16. Mean Square Prediction of Machine Learning Model



	Linear Model	Random Forest	Neural Network	Decision Tree	SVM	Decision Stump
WMC+RFC+LOC Metrics	80.6	77.7	78.5	77	82.5	77.4
All the metrics	74.4	77.7	53.1	79.5	83	80.6

Fig.17. Accuracy Prediction of Machine Learning Model

B. Findings

Pandey N et.al [27] have used a machine learning techniques Naive Bayes, linear discrimination analysis, K-Nearest neighbors, Support Vector Machine, decision tree and random forest to find the high performance from three open source projects JIRA which belongs to APACHE, LUCENE, JACKRABBIT. They derive that Random Forest perform best accuracy of 79% and SVM accuracy was 75%. Singh P.D et.al [28] in their paper have analyzed five machine learning model to predict the software defect prediction. They have taken 7 dataset from NASA Promise repository. They have used KEEL tool and classification technique was used in Artificial Neural Network (NN), Decision Tree (DT), Linear Classifier (LC), Naive Bayes (NB), P article swarm ptimization (PS) machine learning model. It was analyzed that linear classifier outperform as compare to other machine learning model. The accuracy level was 83%. Fig 18 depict that SVM is the best model to predict the software defect as per the experiment done using nine datasets and seven datasets as 83% and 88% respectively and compared with accuracy prediction of different machine learning technique by Pandey N et.al and Singh P.D et.al as 79% and 83% respectively.

VII. Conclusion

The objective of the paper was to find the best machine learning model for software bug prediction. To get the most optimal machine learning model the accuracy, mean square error, R Squared and correlation was computed. It was seen that the maximum accuracy and minimum mean square error was derived from the support vector machine (SVM) machine learning model for finding the software bug prediction.

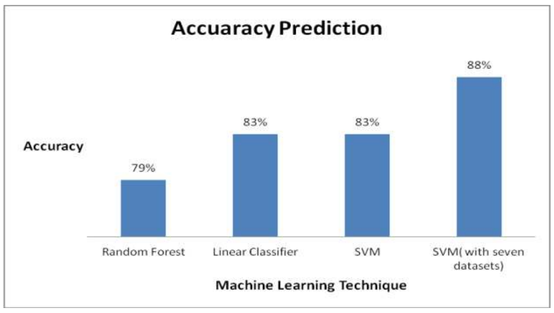


Fig.18. Accuracy prediction in different machine learning technique

References

[1] S. Puranik, P. Deshpande, and K. Chandrasekaran, “A Novel Machine Learning Approach for Bug Prediction,” *Procedia - Procedia Comput. Sci.*, vol. 93, no.September, pp. 924–930, 2016. “doi:10.1016/j.procs.2016.07.271”.

[2] K. O. Elish and M. O. Elish, “Predicting defect-prone software modules using support vector machines,” *Journal of Systems and Software*, vol. 81, no. 5, pp. 649–660, 2008. “doi:10.1016/j.jss.2007.07.040”

[3] M. Dhiauddin, M. Suffian, and S. Ibrahim, “A Prediction Model for System Testing Defects using Regression Analysis,” *Int. J. Soft Comput. Softw. Eng.*, vol. 2, no. 7, pp. 55–68, 2012. “doi:10.7321/jscse.v2.n7.6”

[4] P. S. Rana, H. Sharma, M. Bahattacharya, and A. Shukla, “Journal of Bioinformatics and Computational Biology c Imperial College Press Quality assessment of modeled protein structure using physicochemical properties,” *J. Bioinform. Comput. Biol.*, vol. 13, no. 2, pp. 1–16, 2015. https://doi.org/10.1142/S0219720015500055

[5] Y. Suresh, J. Pati, and S. K. Rath, “Effectiveness of software metrics for object-oriented system,” *Procedia Technology* vol. 6, pp. 420–427, 2012.” doi: 10.1016/j.protcy.2012.10.050”

[6] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, “Choosing software metrics for defect prediction : an investigation on feature selection techniques,” *Software: Practice and Experience* pp. 579–606, 2011.” doi: 10.1002/spe.1043”

[7] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” *R news*, vol. 2, no. December, pp. 18–22, 2002. “doi: 10.1177/154405910408300516” .

[8] M. B. Kursa and W. R. Rudnicki, “Feature Selection with the Boruta Package,” *J. Stat. Softw.*, vol. 36, no. 11, pp. 1–13, 2010.”doi: Vol. 36, Issue 11, Sep 2010 “

[9] Cran.r-project.org, 2018. [Online]. Available: https://cran.r-project.org/web/packages/leaps/leaps.pdf. [Accessed: 21- Feb- 2018].

[10] Y. Suresh, J. Pati, and S. K. Rath, “Effectiveness of software metrics for object-oriented system,” *Procedia technology* vol. 6, pp. 420–427, 2012.”doi: 10.1016/j.protcy.2012.10.050”

[11] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” *R news*, vol. 2, no. December, pp. 18–22, 2002.”doi: 10.1177/154405910408300516”.

[12] Y. Song and Y. Lu, “Decision tree methods: applications for classification and prediction,” *Biostatistics in psychiatry*, vol. 27. pp. 130–135, 2015.”doi: 10.11919/j.issn.1002-0829.215044”

[13] P. Romanski, L. Kotthoff, and M. L. Kotthoff, “Package FSelector: Selecting Attributes,” *Cran*, p. 18, 2016.”

[14] P. A. Selvaraj and P. Thangaraj, “Support Vector Machine for Software Defect Prediction,” *Int. J. Eng. Technol. Res.*, vol. 1, no. 2, pp. 68–76, 2013.

[15] E. Osuna, R. Freund, and F. Girosit, “Training support vector machines: an application to face detection,” *Proceedings of IEEE Computer Society Conference on Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 130–136, 1997. “doi:10.1109/cvpr.1997.609310”

[16] F. E. H. Tay and C. Lijuan, “Application of Support Vector Machines in Financial Time Series Forecasting,” *Omega*, vol. 29, no. 2001, pp. 309–317, 2001. “doi: 10.1016/s0305-0483(01)00026-3”

[17] Y. Song and Y. Lu, “Decision tree methods: applications for classification and prediction,” *Biostatistics in psychiatry*, vol. 27. pp. 130–135, 2015.” doi:10.11919/j.issn.1002-0829.215044”

[18] F. E. H. Tay and C. Lijuan, “Application of Support Vector Machines in Financial Time Series Forecasting,” *Omega*, vol. 29, no. 2001, pp. 309–317, 2001.

[19] E. Rahm and H. H. Do, “Data cleaning: Problems and current approaches,” *IEEE Bull. Data Eng.*, vol. 23, no. 4, pp, 2000.

[20] “Documentation,” *Machine Learning in MATLAB - MATLAB & Simulink - MathWorks India*. [Online]. Available: https://in.mathworks.com/help/stats/machine-learning-in- matlab.html. [Accessed: 03-Jul-2017].

[21] “1.10. Decision Trees¶,” *1.10. Decision Trees — scikit-learn 0.18.2 documentation*. [Online]. Available :http ://

scikit-learn.org/stable/ module s/tree.html. [Accessed: 04-Jul-2017].

[22] S. Kim, "Introduction to Machine Learning for developers," *Algorithmia Blog*, 28-Feb-2017. [Online].Available: https:// blog. algorithmia. com/ introduction -machine-learning- developers/. [Accessed: 05-Jul-2017].

[23] "Random forest," *Wikipedia*, 04-Jul-2017. [Online]. Available:https://en. wikipedia.org /wiki/ Random_ forest. [Accessed: 05-Jul-2017].

[24] S. Ray, S. Bansal, A. Gupta, D. Gupta, and F. Shaikh, "Understanding Support Vector Machine algorithm from examples (along with code)," *Analytics Vidhya*, 13-Sep-2016. [Online]. Available: https://www. analyticsvidhya.com/ blog/2015/10/understaing-support-vector-machine-example-code/. [Accessed: 05-Jul-2017].

[25] E. Rahm and H. Do, "Data cleaning: Problems and current approaches," *Bull. Tech. Comm.*, 2000.

[26] R. K. H. Galvão and A. M.C.U., "Variable Selection," *Compr. Chemom.*, pp. 233–283, 2009.

[27] N. Pandey, D. K. Sanyal, A. Hudait, and A. Sen, "Automated classification of software issue reports using machine learning techniques: an empirical study," *Innov. Syst. Softw. Eng.*, pp. 1- 19, 2017. "doi: 10.1007/s11334-017-0294-1"

[28] P. Deep Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," *2017 7th Int. Conf. Cloud Comput. Data Sci. Eng. - Conflu.*, pp. 775–781, 2017. " doi: 10.1109/ CONFLUENCE .2017. 7943255".

[29] M. Singh and D.S. Salaria. "Software defect prediction tool based on neural network". *International Journal of Computer Applications*. Vol. 70 No. 22. pp- 22-28, 2013."doi: 10.5120/12200-8368".

[30] A. Okutan and O. T. Yildiz, "Software defect prediction using Bayesian networks," Empir. Softw. Eng., vol. 19, no. 1, pp. 154–181, 2014."doi: 10.1007/s10664-012-9218-8"

[31] A. Kaur, K. Kaur, and D. Chopra, "An empirical study of software entropy based bug prediction using machine learning," Int. J. Syst. Assur. Eng. Manag., 2016. "doi: 10.1007/s10664-012-9218-8".

[32] X. Rong, F. Li and Z. Cui. "A model for software defect prediction using support vector machine based on CBA". Int. J. Intelligent Systems Technologies and Applications, Vol. 15, No. 1, pp- 19-34. 2016. "doi: 10.1504/ijista.2016.076102".

Authors' Profiles

Tripti Lamba is pursuing Ph.D from JIMS, Jaipur. She has done her M. Tech from Punjabi Univ, Patiala. She has 15 yrs of experience. Currently working with Institute of Information and Technology (IITM)as an Asst. Professor –IT. Her areas of interest are web technologies and data mining. She has published five papers in reputed International / National journals.

Dr. Kavita Ph.D (Computer Science) received her M.C.A degree in computer science from Modi Institute of technology and Science Lakshmangarh, Sikar. Presently working as Associate Professor at Jyoti Vidyapeeth University Jaipur. She has eleven years of teaching experience in the field of Computer Science and supervising research scholars in the field of E-commerce, Mobile Commerce, Data Mining, big data, Cloud computing etc.

Dr. A.K.Mishra has published several research papers in the reputed 'National and International Journals'.

A.K.MISHRA is a Principal Scientist in Indian Agricultural Research Institute, Pusa , New Delhi, India. He is an IT expert with more than 20 yrs. of experience which includes application designing, implementation and management of ICT based projects. He has experience in implementing projects in the domain of Knowledge and e-Resource management. He has published 25 papers in reputed International / National journals. His area of interest are bioinformatics, Web technologies, Software Engineering, IT in Agriculture and Rural Development.

How to cite this paper: Tripti Lamba, Kavita, A.K.Mishra, "Optimal Machine learning Model for Software Defect Prediction", International Journal of Intelligent Systems and Applications(IJISA), Vol.11, No.2, pp.36-48, 2019. DOI: 10.5815/ijisa.2019.02.05