Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change-Proneness

Lov Kumar
National Institute of Technology (NIT)
Rourkela, India
lovkumar505@gmail.com

Santanu Kumar Rath
National Institute of Technology (NIT)
Rourkela, India
skrath@nitrkl.ac.in

Ashish Sureka
ABB Corporate Research
Bangalore, India
ashish.sureka@in.abb.com

ABSTRACT

Change-prone classes or modules are defined as software components in the source code which are likely to change in the future. Change-proneness prediction are useful to the maintenance team as they can optimize and focus their testing resources on the modules which have a higher likelihood of change. The quality of change-proneness prediction model can be best assessed by the use of software metrics that are considered to design the prediction model. In this work, 62 software metrics with four metrics dimensions, including 7 size metrics, 18 cohesion metrics, 20 coupling metrics, and 17 inheritance metrics are considered to develop a model for predicting change-proneness modules. Since the performance of the change-proneness model depends on the source code metrics, they are used as input of the change-proneness model. We also considered five different types of feature selection techniques to remove irrelevant feature and select best set of features.

The effectiveness of these set of source code metrics are evaluated using eight different machine learning algorithms and two ensemble techniques. Experimental results demonstrates that the model developed by considering selected set of source code metrics by feature selection technique as input achieves better results as compared to considering all source code metrics. The experimental results also ravel that the change-proneness model developed by using coupling metrics achieved better performance as compared other dimension metrics such as size metrics, cohesion metrics, and inheritance metrics.

Keywords

Change-Proneness, Source Code Metrics, Feature Selection Techniques, Classification Technique, Ensemble Techniques

1. INTRODUCTION

Change-prone classes or modules are defined as software components in the source code which are likely to change in the future. Prediction and early identification of such components are useful to the maintenance team as they can optimize and focus their testing resources on the modules which have a higher likelihood of change. Prediction of change prone components is an area which has attracted several researchers attention. Building effective and accurate change-proneness predictive models is a technically challenging problem. Change-proneness prediction models are generally developed using structural measurement of software (software metrics) i.e, size, cohesion, coupling, and inheritance [2][4][17][1][21]. Source code metrics are used to measure the internal structure of software system such as complexity, coupling, cohesion, inheritance, and size. In this work, 62 software metrics with four metrics dimensions, including 7 size metrics, 18 cohesion metrics, 20 coupling metrics, and 17 inheritance metrics are considered to develop a model for predicting change-proneness modules.

Since a change-proneness prediction model is based on source code metrics, the selection of the suitable set of source code metrics becomes an integral component of the model development process. Selection of right set of features or metrics is an important data pre-processing task in different application of data mining and machine learning . In the present work, five different types of features selection techniques such as univariate logistic regression analysis, gain ratio feature evaluation, information gain feature evaluation, principal component analysis (PCA), and rough set analysis (RSA) are considered to to validate the source code metrics and identify suitable set of source code metrics with an aim to reduce irrelevant metrics and improve the performance of change-proneness prediction model. The effectiveness of these set of source code metrics are evaluated using eight different learning algorithms used in our study are: Logistic Regression (LOGR), Naive Bayes Classifier (NBC), Extreme Learning Machine (ELM) with linear (LIN), polynomial (PLY) and Radial Basis Function (RBF) kernels, Support Vector Machine (SVM) with linear (LIN), RBF and Sigmoid kernel (SIG) and two ensemble techniques such as Best-in-Training (BTE) and Majority Voting (MV) [16][27][26].

The research aim of the work presented in this paper is to investigate the application of 62 source code metrics, 3 different feature extraction or selection methods and 2 different ensemble methods for predicting change proneness. The number and type of source code metrics, feature selection methods and machine learning techniques in our study is unexplored and forms the novel and unique research contributions of our work.

To the best of our knowledge, the study presented in this

ISEC '17, February 05-07, 2017, Jaipur, India
© 2017 ACM. ISBN 978-1-4503-4856-0/17/02...$15.00
DOI: http://dx.doi.org/10.1145/3021460.3021461

paper is the first study on validation of source code metrics and developed change-proneness prediction models. The summary of our proposed work is as follows:

1 Four different source code metrics dimensions such as size metrics (7), cohesion metrics (18), coupling metrics (20), and inheritance metrics (17) are considered to evaluate the predictive effectiveness of source code metrics for change-proneness prediction.

2 Five different types of features selection techniques such as univariate logistic regression analysis, gain ratio feature evaluation, information gain feature evaluation, principal component analysis (PCA), and rough set analysis (RSA) are used to select right set of source code metrics.

3 Eight different learning algorithms used in our study are: LOGR, NBC, ELM-LIN, ELM-PLY, ELM-RBF, SVM-LIN, SVM-RBF, SVM-SIG are considered to develop a model to predict change-proneness of OO software.

4 Two different types of *ensemble methods* are also applied to come up with better performance as compared to the individual models.

2. RELATED WORK

There are several source code metrics and perdition models proposed by different authors to predict the change-prone of the software [14][15]. The methods used to develop change-prone prediction model vary form regression analysis to simple machine learning algorithm such as neural networks etc. We elaborate few important studies here on uses of source code metrics and their application for developing change-prone prediction model.

Henry and Kafura considered correlation analysis for measuring the correlation between changeability i.e., number of changed source lines in the Unix operating system and source code metrics [17]. They found that the source code metrics are strongly correlated with changeability. They defined these source code metrics using information flow among the system components.

Ruchika and Anuradha studied the relationship between object oriented metrics and maintainability i.e., changeability of software [20]. They computed source code metrics for old and new versions for both applications and analyzed against modifications made in every class. They conclude that the software professionals can use Object-Oriented source code metric to predict the maintainability of software system.

Ah-Rim Han et al. proposed a metrics to measure behavioral dependency by considering structural and behavioral information of UML 2.0 design models [10]. They observed that the prediction of Model-based change-proneness helps to make high-quality software by exploiting design models from the earlier phase of the software development process. They concluded that the proposed metrics is a useful indicator and can be complementary to existing object-oriented metrics for improving the accuracy of change-proneness prediction when the system contains high degree of inheritance relationships and polymorphism.

Hongmin Lu et al. considered statistical meta-analysis techniques to investigate the ability of sixty Object-Oriented source code metrics to predict change-proneness [19]. In their work, they defined change-prone as "a class which is changed in the next version of a system is called change-prone and not change-prone otherwise". They considered four different metrics dimension such as cohesion, coupling, size, and inheritance for investigating the Object-Oriented source code metrics. They concluded that prediction model developed by considering cohesion and coupling source code have lower predictive ability compared to size metrics. They also concluded that inheritance metrics have a poor ability to discriminate between change-prone and not change-prone classes.

Yuming and Hareton considered multiple adaptive regression splines (MARS) modeling technique to build software maintainability prediction models using the software metrics [29]. These software metrics data collected from two different object-oriented systems. They evaluate the performance of MARS models and compared using regression tree models, artificial neural network models, multivariate linear regression models, and support vector models. They observed that model developed using MARS more accurately predict maintainability than the other four typical modeling techniques, and that for the other system MARS is as accurate as the best modeling technique.

Limitations of Existing Work: From literature survey, we observed that authors considered different set of source code metrics to predict change-proneness of Object-Oriented software. This shows that the change-proneness prediction model performance depends on the source code metrics which have been considered as input to develop a model. Selection of suitable set of feature is an important step of data analysis in various domains. The application of feature selection techniques in different domains have been reported in literature [7] [8] [5] [28] [3] [12] [11]. The limitations of these studies may be identified that the comparative analysis are not performed between performance of model developed using all features and selected set of features. In this work, five different types of features selection techniques such as univariate logistic regression analysis, gain ratio feature evaluation, information gain feature evaluation, principal component analysis (PCA), and rough set analysis (RSA) have been considered to find right subset of software metrics. In this work, we also considered four different metric dimensions such as size, cohesion, coupling and inheritance source code metrics to evaluate the predictive effectiveness of these different dimension metrics on change-proneness of Object-Oriented software. The effectiveness of these set of source code metrics are evaluated using eight different learning algorithms used in our study are: LOGR, NBC, ELM-LIN, ELM-PLY, ELM-RBF, SVM-LIN, SVM-RBF, SVM-SIG and two ensemble techniques such as Best-in-Training (BTE) and Majority Voting (MV).

3. EXPERIMENTAL DATASET

3.1 Case Study

For our experiments, we use two different version of Eclipse software application (Eclipse version. 2.0 [1] and Eclipse ver-

[1]http://archive.eclipse.org/eclipse/downloads/drops/R-2.0-200206271835/eclipse-sourceBuild-srcIncluded-2.0.zip

sion 2.1 [2]). We use Eclipse as a case-study to analyze the effectiveness of the proposed approach as Eclipse is a long-running, widely-used, publicly available, large and complex open-source project. We notice that in Eclipse application there are several classes which are defined in different Java files but have the same full name (i.e. package name followed by the class name). Hence, many classes have the same full name but different program implementations. scitool for Java [3] cannot distinguish between such cases and can introduce bias. This biases the computations of the Object Oriented metrics as well as metrics such as the class size. For elimination of such bias, we excluded all *redundant* Java files such that each class has one and only one implementation. In addition, we also eliminated all Java files that contain statements starting with %, as Understand for Java tool[4] cannot correctly process them. After eliminating and removing all irrelevant Java files, we compute the source code metrics value for only those Java file which appear in both the versions i.e., Eclipse 2.0 and Eclipse 2.1. We use Perl tool to compute the change-proneness module between two version of Eclipse software [5] [19].

3.2 Source Code Metrics

Code Metrics Classification: We classify the 62 source code metrics used in our experiments into 4 categories: cohesion, coupling, inheritance and size. In our study we use 19 cohesion metrics, 19 coupling metrics, 17 inheritance metrics and 7 size metrics. The information flow among various program components in the object oriented software implementation is measured using coupling and there are several metrics such as Data Abstraction Coupling (DAC) and Coupling between Objects (CBO) to measure coupling [23]. Cohesion is defined as the degree or extent to which various elements in a design unit such as packages and classes are related to each-other and is measured using several metrics such as Lack of Cohesion in Methods (LCOM) and Information-Flow based Cohesion (ICH) [22]. Inheritance metrics such as Average Inheritance Depth (AID) and Class-to-Leaf Depth (CLD) metric measure quality and complexity of class inheritance hierarchies [24]. Size metrics such as (Lines of Code) LOC determine the size of the program code [18].Table 1 displays the 62 source code metrics. The 19 cohesion metrics are from COA to TCC (separated using the gray cell shading). As shown in Table 1, the 19 coupling metrics are from ACAIC to RFC, 17 inheritance metrics from AID to SPD and 7 size metrics from NA to SLOC.

Code Metrics Descriptive Statistics: Table 1 displays the descriptive statistics computed by us for the 62 source code metrics on our experimental dataset. Table 1 reveals substantial variation or dispersion in the values of 62 source code metrics which shows wide variability in the structure and size of program code and elements. Table 1 reveals that the Source Lines of Code (SLOC) value varies from a minimum of 4 to a maximum of 3669. The mean value of 115.431 for SLOC means that a large number of Classes have SLOC more than 100.

Metrics Correlation Analysis: We compute the association between 62 metrics consisting of dependent and independent variables using the Pearson's correlations coefficient (r). The coefficient of correlation r measures the strength and direction of the linear relationship between two variables. Figure 1 displays our experimental results on correlation analysis between the 62 metrics. In Figure 1, a Black circle represents an r value between 0.7 and 1.0 or between -0.7 and -1.0 indicating a strong positive or negative linear relationship respectively. A white circle r value between 0.3 and 0.7 or -0.3 and -0.7 indicating a weak positive or negative linear relationship respectively. A blank cell represents no linear relationships between the two variables. For example, based on Figure 1, we infer that there is a strong positive linear relationship between LCOM1 and seven other variables LCOM2, LCOM3, LCOM4, NMA, NAIMP, NUMPA and SLOC. On the other hand, we observe a weak linear relationship between CAMC and DCD as well as CBO and ICH. Figure 1 reveals association between different suite of metrics and not just associations between metrics within the same suite. For example, metrics such as NA, NAIMP, NM, NMIMP and NUMPA are part of the size metrics and they have strong as well as weak correlations with several coupling metrics OCAEC, NIHICP, MPC and IHICP.

3.3 Effectiveness of Metrics

In this work, ten different set of source code metrics (all metrics (AM), cohesion metrics (CHM), coupling metrics (CPM), size metrics (SM), inheritance metrics (IHM), selected set of metrics using gain ratio feature (GRS), selected set of metrics using information gain (IGS), selected set of metrics using univariate logistic regression analysis (LCS), extracted set of metrics using principal component analysis (PCA), selected set of metrics using rough set analysis (RSA)) are considered as input to develop change-proneness prediction model. Table 2 show the independent and dependent variables used for change-proneness model development.

4. FEATURE SELECTION METHODS

We apply 4 different feature selection methods as the number of dimensions in our dataset is high. Our objective is to eliminate some of the irrelevant and redundant original variables to increase the training speed and accuracy of the classifier. We apply a filter approach for feature selection which precedes the classifier design and is also independent of the learning algorithm. We chose the filter approach for feature selection as we build 10 different classifiers to which the subset of features from each feature selection is provided as input. The four feature selection techniques that we use in our study are **(1)** Univariate Logistics Regression **(2)** Principal Component Analysis **(3)** Information Gain and Gain Ratio **(4)** Rough Set Analysis. The 4 methods provides us guidance on selection a subset of the original features which are useful in developing a good estimator or predictor of change-proneness.

Univariate Logistic Regression (ULR): We apply univariate logistic regression to evaluate the relationship and individual effect of each of the 62 source code metrics on the change-proneness of classes. Our research objective is to identify metrics which are significantly, moderately and not related to the change-proneness of classes. Univariate logis-

[2]http://archive.eclipse.org/eclipse/downloads/drops/R-2.1-200303272130/eclipse-sourceBuild-srcIncluded-2.1.zip
[3]http://www.scitools.com
[4]https://scitools.com/
[5]https://www.perl.org/get.html

Table 1: Descriptive Statistics for 62 Source Code Metrics [Cohesion: From COA to TCC, Coupling: From ACAIC to RFC, Inheritance: From AID to SPD, Size: From NA to SLOC]

Metric	Max	Min	Mean	Median	Std Dev	Metric	Max	Min	Mean	Median	Std Dev
COA	110	0	0.4	0	3.636	**OCAEC**	500	0	1.47	0	13.1
CAMC	1	0.014	0.221	0.179	0.15	**OCAIC**	188	0	3.552	2	6.934
CO	1	-2	0.122	0.089	0.461	**OCMEC**	1156	0	4.065	0	32.189
DCD	1	0	0.54	0.5	0.379	**OCMIC**	377	0	8.606	5	15.234
DCI	1	0	0.617	0.714	0.391	**OMMEC**	4319	0	24.329	3	125.093
ICH	2976	0	21.051	4	85.856	**OMMID**	643	0	24.01	8	44.389
LCC	1	0	0.604	0.672	0.397	**RFC**	2240	1	265.798	103	337.911
LCOM1	171850	0	217.239	23	3018.651	**AID**	7	0	1.265	1	1.343
LCOM2	166390	0	185.78	13	2893.3	**CLD**	7	0	0.309	0	0.776
LCOM3	492	1	6.78	4	12.283	**DIT**	7	0	1.265	1	1.343
LCOM4	282	1	3.236	2	6.176	DP	420	0	4.82	1	16.778
LCOM5	2	0	0.771	0.833	0.28	**DPA**	129	0	2.286	0	5.233
NHD	1	0	0.651	0.69	0.205	**DPD**	415	0	2.534	0	15.874
NEWCO	1	0	0.369	0.306	0.274	**NMA**	596	0	11.429	7	18.455
NEWLCOM5	1	0	0.333	0.267	0.243	**NMI**	596	0	23.809	6	42.686
OCC	1	0	0.655	0.714	0.31	**NMO**	124	0	1.988	0	4.316
PCC	1	0	0.534	0.563	0.332	**NOA**	7	0	1.265	1	1.343
SNHD	1	-1	-0.187	0	0.527	**NOC**	155	0	0.739	0	3.967
TCC	1	0	0.53	0.5	0.383	**NOPD**	381	0	1.606	0	10.654
ACAIC	4	0	0.036	0	0.238	**NOP**	1	0	0.626	1	0.484
ACMIC	17	0	0.128	0	0.718	**SIX**	1.596	0	0.094	0	0.156
AMMIC	321	0	5.434	1	13.216	**SP**	110	0	0.665	0	3.74
CBI	3306815	0	3658	0	67233	**SPA**	10	0	0.264	0	0.742
CBO	164	0	13.14	9	14.39	**SPD**	110	0	0.4	0	3.636
DAC	188	0	3.65	2	6.987	**NA**	1274	1	16.28	9	34.014
DCAEC	19	0	0.027	0	0.485	**NAIMP**	1053	1	6.711	3	21.117
DCMEC	72	0	0.137	0	2.237	**NM**	699	1	37.226	23	48.661
ICP	1404	0	62.725	24	115.653	**NMIMP**	596	1	13.417	9	19.1
IHICP	1001	0	10.653	2	30.345	**NUMPA**	1753	1	15.029	8	37.018
MPC	643	0	29.533	12	50.858	**STMTS**	5200	6	156.463	80	262.763
NIHICP	1404	0	52.072	18	101.99	**SLOC**	3669	4	115.431	59	191.54

Table 2: Effectiveness of Metrics

Analysis	Dependent Variable	Independent Variable
A1	Change-proneness	All metrics
A2	Change-proneness	Cohesion metrics
A3	Change-proneness	coupling metrics
A4	Change-proneness	Size metrics
A5	Change-proneness	Inheritance metrics
A6	Change-proneness	selected set of metrics using gain ratio feature (GRS)
A7	Change-proneness	selected set of metrics using information gain (IGS)
A8	Change-proneness	selected set of metrics using univariate logistic regression analysis (LCS)
A9	Change-proneness	Extracted feature attributes using PCA
A10	Change-proneness	Reduced feature attributes using RSA

tic regression helps in computing the percent or extent of variance (a predictor of statistical relationship between two variables) in the dependent variable explained by the independent variables. The 62 source code metrics represents independent variables and the dependent variable represents the change value for each class (can take only of the two values: changed or not changed). Table 3 reveals that the p-value of metrics OCC, PCC, DCAEC, DCMEC, OCAEC, OCMEC, NOC and NOPD is greater than the commonly used alpha threshold or level of 0.05 and hence they are not statistically significant predictors. From the p-values, we infer that we should remove metrics OCC, PCC, DCAEC, DCMEC, OCAEC, OCMEC, NOC and NOPD from model building. We observe that 54 of the 62 metrics have a low p-value value ranging between 0 and 0.05 and hence they are useful predictors for our change-prone estimator. Table 3 reveals different regression coefficients for various predictor variables. We observe that CAMC, NHD, NEWLCOM5 and SIX (gray colored cells) have relatively high regression coefficients which means that the mean change in the change-prone variable (response variable) is high for one unit change in the respective dependent variable representing the source code metric.

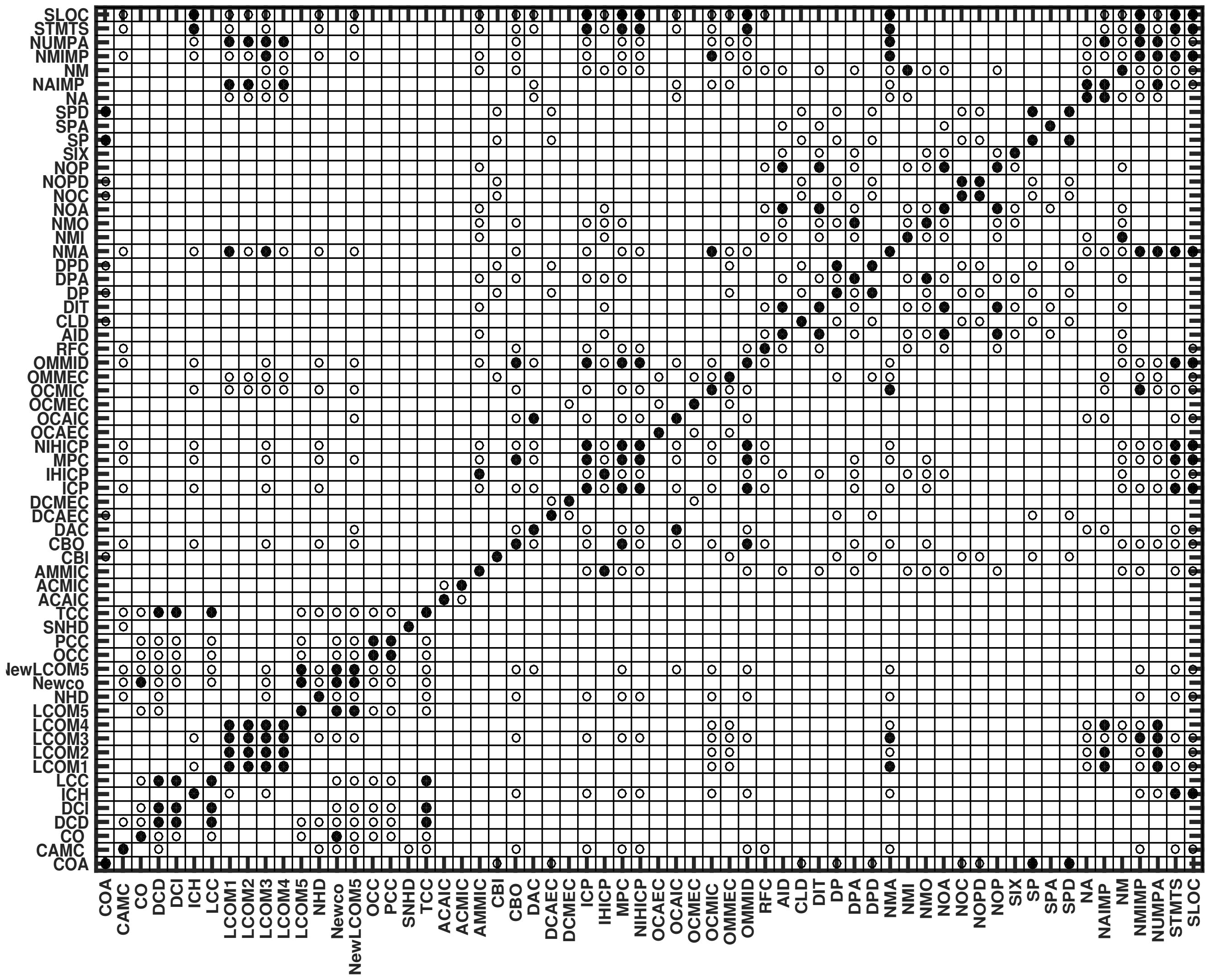


Figure 1: Correlation Matrix between 62 Source Code Metrics

Principal Component Analysis (PCA): Since there are a large number of independent variables (62 source code metrics), we apply PCA to reduce the number of variables based on variable pairwise correlation and variance for easier interpretation and analysis. Figure 1 reveals that there are correlations or multi-collinarity between several source code metrics and hence it is possible to reduce the dimensions. Table 4 shows the eigenvalues of the 14 principal components in the decreasing order of eigenvalues and the proportion of variance (in terms of percentage variance) explained by the 14 principal components. Table 4 shows only those principal components whose eigenvalue is greater than 1.0. Table 4 also reveals the correlations between the 14 principal components and the 62 to source code metrics. For example, experimental results reveal that AID, DIT, NMI, NOA, NOP and SIX are most strongly correlated with the $PC2$ component. We infer from PCA procedure that AID, DIT, NMI, NOA, NOP and SIX vary together. As shown in Table 4, our investigation using PCA procedure provides us guidance on dimensionality reduction, metric selection and the identification of dependence variables to be used for predictive model building.

Gain Ratio (GR) and Information Gain (IG): Information gain (IG) assesses the predictive power of a feature by computing the information gain with respect to the target class and is calculated by evaluating the difference between changes in entropy of prior state and new state [9]. Gain ratio (GR) is a modification of information gain measure and is calculated as the ratio between the information gain and the intrinsic value. We apply the procedure of selecting top $\lceil \log_2 n \rceil$ metrics out of n metrics [9]. In our study $n = 62$ and hence we select the top 6 metrics. The 6 metrics selected using gain ratio are ICP, NIHICP, MPC, OMMID, STMTS, NOPD and the 6 metrics selected using information gain are CBO, MPC, ICP, OMMID, STMTS, NIHICP. Table 5 displays the information gain and gain ratio values for all the 62 metrics. Table 5 reveals the selected features represented using shaded gray cell.

Table 3: Univariate Logistic Regression (ULR) Procedure Experimental Results

Metrics	Coff.	p-val	Metrics	Coff.	p-val	Metrics	Coff.	p-val	Metrics	Coff.	p-val	Metrics	Coff.	p-val
COA	0.044	0.028	**NEWCO**	-1.436	0	**DCMEC**	0.019	0.326	**AID**	0.362	0	**NOP**	0.729	0
CAMC	-4.399	0	**NEWlCOM5**	-2.434	0	**ICP**	0.021	0	**CLD**	0.139	0.004	**SIX**	2.22	0
CO	-0.089	0.246	**OCC**	-0.218	0.056	**IHICP**	0.03	0	**DIT**	0.362	0	**SP**	0.134	0
DCD	-0.694	0	**PCC**	0.015	0.886	**MPC**	0.046	0	**DP**	0.04	0	**SPA**	0.521	0
DCI	-0.433	0	**SNHD**	-0.577	0	**NIHICP**	0.025	0	**DPA**	0.151	0	**SPD**	0.044	0.028
ICH	0.034	0	**TCC**	-0.728	0	**OCAEC**	0.01	0.083	**DPD**	0.013	0.001	**NA**	0.051	0
LCC	-0.489	0	**ACAIC**	0.555	0.003	**OCAIC**	0.218	0	**NMA**	0.067	0	**NAIMP**	0.111	0
LCOM1	0.004	0	**ACMIC**	0.157	0.014	**OCMEC**	0.001	0.492	**NMI**	0.013	0	**NM**	0.021	0
LCOM2	0.005	0	**AMMIC**	0.076	0	**OCMIC**	0.08	0	**NMO**	0.185	0	**NMIMP**	0.084	0
LCOM3	0.146	0	**CBI**	0	0	**OMMEC**	0.002	0.006	**NOA**	0.362	0	**NUMPA**	0.064	0
LCOM4	0.155	0	**CBO**	0.128	0	**OMMID**	0.055	0	**NOC**	0.015	0.203	**STMTS**	0.009	0
LCOM5	1.268	0	**DAC**	0.21	0	**RFC**	0.003	0	**NOPD**	0.013	0.022	**SLOC**	0.012	0
NHD	3.103	0	**DCAEC**	0.027	0.728									

Table 4: Principle Component Analysis (PCA) Procedure Experimental Results

PC	Eigenvalue	% Variance	Cumulative	Correlated Metrics
PC1	7.48	12.076	12.076	ICH, LCOM3, LCOM4, NEWLCOM5, AMMIC, CBO, ICP, IHICP, MPC, NIHICP, OCMIC, OCMMEC, OMMID, RFC, DPA, NMA, NMO, NA, NAIMP, NM, NMIMP, NUMPA, STMTS, SLCOC
PC2	6.47	10.448	22.523	AID, DIT, NMI, NOA, NOP, SIX
PC3	5.67	9.149	31.667	COA, DCD, OCC, CBI, DCAEC, CLD, DP, DPD, NOC, NOPD, SP
PC4	5.32	8.589	40.256	CO, DCI, LCC, NEWCO, PCC, TCC
PC5	4.01	6.477	46.734	LCOM1, LCOM2
PC6	3.89	6.274	53.007	DCMEC, OCAEC, OCMEC
PC7	2.61	4.207	57.215	CAMC
PC8	2.47	3.985	61.2	DAC, OCAIC
PC9	2.22	3.583	64.783	LCOM5
PC10	2.14	3.456	68.239	NHD, ACMIC
PC11	2.13	3.448	71.687	SNHD
PC12	2.12	3.428	75.115	ACAIC
PC13	1.9	3.209	78.324	SPA
PC14	1.55	2.5	80.824	SPD

Rough Set Analysis (RSA): We apply Rough Set Analysis (RSA) based approach for feature selection and extracting a subset of attributes from the original set consisting of 62 source code metrics with the objective of removing irrelevant and redundant features. Hence the goal of data pre-processing behind applying RSA is the same as ULR and PCA. We apply the procedure described by Thangavel et al. on application of clustering for feature selection based on rough set theory approach [25]. We apply the k-means clustering algorithm and then formulate the decision table using the clustered data as the decision variable followed by the application of reduction algorithms as described in Thangavel et al [25]. Table 6 displays the group range for the clustered data and the subset of features selected as output to the RSA procedure. Table 6 reveals that the dimensionality has reduced from 62 original metrics to 41 metrics.

5. RESEARCH METHOD

In this paper, eight different learning algorithms such as Logistic Regression (LOGR), Naive Bayes Classifier (NBC), Extreme Learning Machine (ELM) with linear (LIN), polynomial (PLY) and Radial Basis Function (RBF) kernels, Support Vector Machine (SVM) with linear (LIN), RBF and Sigmoid kernel (SIG) and two ensemble techniques such as Best-in-Training (BTE) and Majority Voting (MV) are considered to develop a change-proneness model. Logistic Regression (LOGR), and Naive Bayes Classifier (NBC) most frequently used classification methods.

ELM is a latest technology which is well adopted for its excellent performance in small applications and also in bigger dataset application. The ELM find out the hidden nodes of single hidden layer feed forward neural network in general unlike neural which is randomly generated and capability of such single layer feed forward neural network is guaranteed. In this paper, there are three types of kernels have been used to map the function in high dimensional space i.e. Linear kernel, Polynomial kernel, Radial basis function (RBF) kernel.

In recent year, SVM models have seen an explosion of interest, and their applicability across a wide range of problem domains [29]. Support Vector Machine(SVM) is based on the theory of statistical learning. The application of SVM are many, such as analyzing data and recognizing patterns which are used for classification and regression analysis. In this study, Support Vector Machine (SVM) method with linear kernel, sigmoid kernel, and radial basis function kernel are considered while developing a model to predict change-profess.

In ensemble of classification models, we have considered the outputs of all its individual constituent classification models i.e., base learners are assigned a certain priority level to each classification model and finally compute the final output with the help of some combination rules. In the present work, we have considered two different ensemble methods such as Best Training Ensemble (BTE) method, and Majority Voting Ensemble (MVE) method to develop change-proneness model [6].

Best Training Ensemble (BTE) Best Training Ensemble (BTE) method takes the advantage of the fact that each

Table 5: Gain Ratio (GR) and Information Gain (IG) Based Feature Selection Experimental Results

Metrics	Gain-Ratio	Info-Gain	Metrics	Gain-Ratio	Info-Gain	Metrics	Gain-Ratio	Info-Gain	Metrics	Gain-Ratio	Info-Gain
COA	0.01	0.004	**NEWCO**	0.027	0.042	**DCMEC**	0	0	**AID**	0.022	0.035
CAMC	0.046	0.088	**NEWLCOM5**	0.04	0.08	**ICP**	0.084	0.167	**CLD**	0	0
CO	0.023	0.063	**OCC**	0.019	0.021	**IHICP**	0.037	0.065	**DIT**	0.022	0.035
DCD	0.018	0.017	**PCC**	0.013	0.015	**MPC**	0.077	0.171	**DP**	0.027	0.042
DCI	0.013	0.022	**SNHD**	0.029	0.039	**NIHICP**	0.078	0.152	**DPA**	0.029	0.043
ICH	0.042	0.081	**TCC**	0.019	0.018	**OCAEC**	0.005	0.005	**DPD**	0.01	0.004
LCC	0.015	0.014	**ACAIC**	0.01	0.002	**OCAIC**	0.038	0.069	**NMA**	0.037	0.055
LCOM1	0.05	0.094	**ACMIC**	0	0	**OCMEC**	0	0	**NMI**	0.027	0.04
LCOM2	0.048	0.089	**AMMIC**	0.038	0.065	**OCMIC**	0.027	0.055	**NMO**	0.028	0.041
LCOM3	0.035	0.078	**CBI**	0.033	0.011	**OMMEC**	0	0	**NOA**	0.022	0.035
LCOM4	0.026	0.03	**CBO**	0.068	0.176	**OMMID**	0.072	0.156	**NOC**	0	0
LCOM5	0.033	0.085	**DAC**	0.039	0.07	**RFC**	0.06	0.141	**NOPD**	0.069	0.003
NHD	0.048	0.088	**DCAEC**	0	0	**NOP**	0.022	0.021	**SIX**	0.028	0.028
SP	0.022	0.017	**SPA**	0.022	0.014	**SPD**	0.01	0.004	**NA**	0.041	0.089
NAIMP	0.032	0.062	**NM**	0.045	0.093	**NMIMP**	0.044	0.086	**NUMPA**	0.039	0.088
STMTS	0.071	0.152	**SLOC**	0.06	0.148						

Table 6: Rough Set Analysis (RSA) Feature Selection Experimental Results

(a) Group and Range

Group	Range
LOW	$[\text{MIN(metrics)}, \frac{C1+C2}{2})$
MEDIUM	$[\frac{C1+C2}{2}, \frac{C2+C3}{2})$
HIGH	$[\frac{C2+C3}{2}, \frac{C3+C4}{2}]$
VERY HIGH	$[\frac{C3+C4}{2}, \text{MAX(metrics)}]$

(b) Reduced Feature Subset

Selected Metrics
CAMC, CO, DCD, DCI, ICH, LCC, LCOM3, LCOM4, LCOM5, NHD, NEWCO, NEWLCOM5, OCC, PCC, SNHD, ACMIC, AMMIC, CBO, ICP, IHICP, MPC, NIHICP, OCAIC, OCMEC, OCMIC, OMMEC, OMMID, RFC, DP, DPA, NMA, NMI, NOA, SIX, SP, NA, NAIMP, NM, NUMPA, STMTS, SLOC

classifiers have different performance across the used dataset partitions. Amongst these, we select the best model in training dataset based on certain performance parameters. In our work, accuracy is considered as a performance parameter to select best trained classifier. Algorithm 1 is used to compute the ensemble output (E_{out}).

Algorithm 1 Best Training Ensemble (BTE) Method

```
1: Select Data with N Number of features.
2: Select M number of classification models.
3: Select K for K-fold cross validation.
4: for each $k \in K$ fold do
5:     for each $m \in M$ model do
6:         Train model $m$ on the training data of k-fold.
7:         Apply model $m$ on the training data of k-fold
8:         Compute training performance of model $m$ ($P_m$) based on certain performance parameter
9:         Store the value of $P_m$
10:    end for
11:    Select best model $M_b \in M$ model based on performance $P_m$
12:    for each $n \in N_{test}$ number of test data for fold $k$ do
13:        $E_{out}$ = Result of model $M_b$ on testing data $n$
14:    end for
15: end for
```

Majority Voting Ensemble (MVE) Method In Majority Voting Ensemble (MVE) method, we have considered the output of each classifier on test data and the ensemble output (E_{out}) is the majority category classified by the base classifier. Algorithm 2 is used to compute the ensemble output (E_{out}).

6. EXPERIMENTAL SETUP AND RESULTS

Figure 2 displays our research framework and methodology. The framework consists of multiple steps. As shown in Figure 1, we first considered dataset containing source code metrics and change-proneness classes of Object-Oriented software. We separate the source code metrics into four different categories i.e., cohesion metrics (CHM), coupling metrics (CPM), size metrics (SZ) and inheritance metrics (IHM) based on their measurement properties. We apply five different feature selection techniques for the purpose of dimensionality reduction and removing irrelevant features. These 10 different set of source code metrics are considered as input to develop a model using eight different learning algorithms and two ensemble techniques. We create 5 set of source code metrics, five different feature selection techniques and 10 different classifier and evaluate the performance of all

Algorithm 2 Majority Voting Ensemble (MVE) Method

```
1: Select Data with N Number of features.
2: Select M number of classification models.
3: Select K for K-fold cross validation.
4: for each $k \in K$ fold do
5:     for each $m \in M$ model do
6:         Apply model $m$ on the training data of k-fold
7:         Compute the output of trained model $m$ on testing data of k-fold
8:         Store the value of output for testing data of the trained model
9:     end for
10:    for each $n \in N_{test}$ number of test data for fold $k$ do
11:        Count the number of models predicting a category
12:        $E_{out}$ = the category which has the maximum count on testing data $n$
13:    end for
14: end for
```

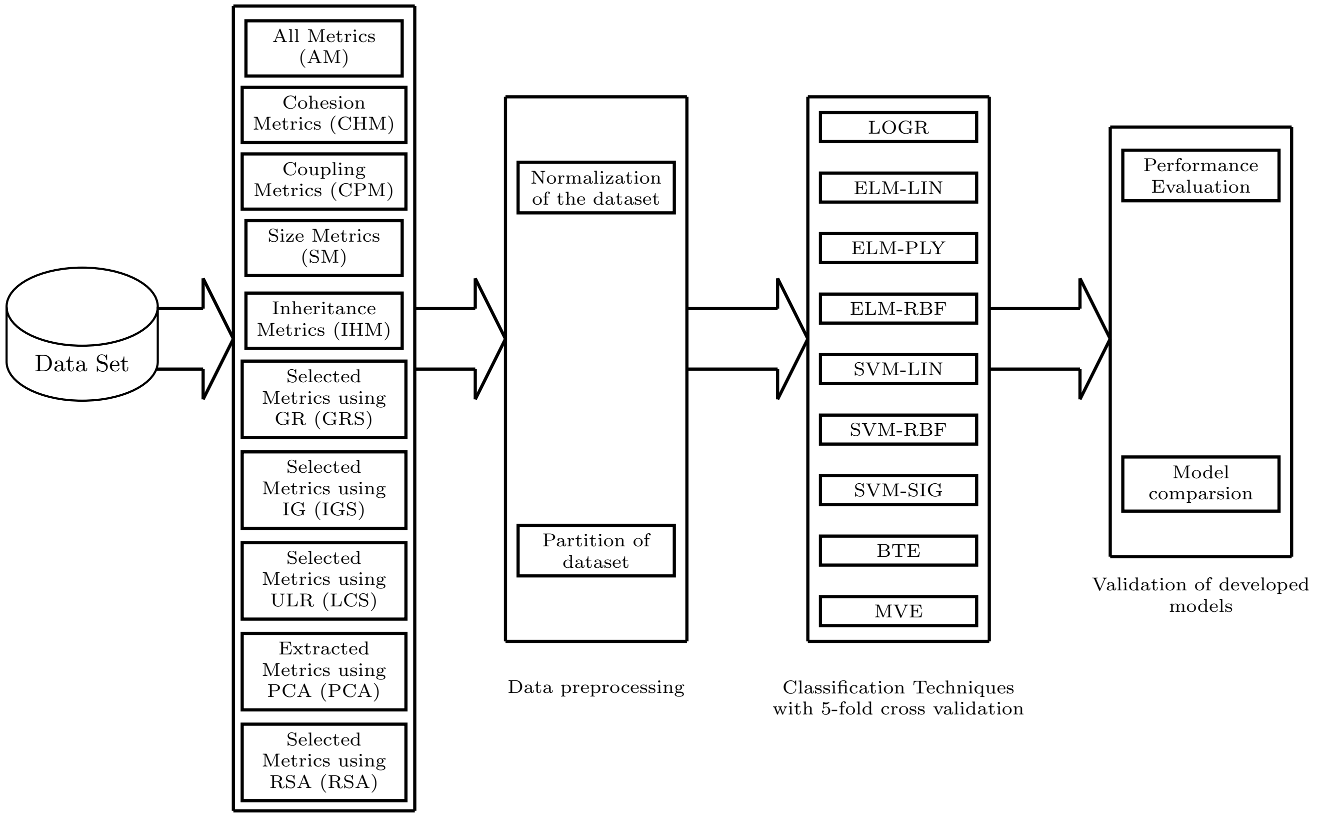

All Metrics (AM)
Cohesion Metrics (CHM)
Coupling Metrics (CPM)
Size Metrics (SM)
Inheritance Metrics (IHM)
Selected Metrics using GR (GRS)
Selected Metrics using IG (IGS)
Selected Metrics using ULR (LCS)
Extracted Metrics using PCA (PCA)
Selected Metrics using RSA (RSA)
Data Set
Normalization of the dataset
Partition of dataset
Data preprocessing
LOGR
ELM-LIN
ELM-PLY
ELM-RBF
SVM-LIN
SVM-RBF
SVM-SIG
BTE
MVE
Classification Techniques with 5-fold cross validation
Performance Evaluation
Model comparsion
Validation of developed models

Figure 2: Framework of Proposed work

the combinations resulting in a comprehensive and in-depth experimental evaluation.

We apply the standard technique of 10 fold cross-validation for the purpose of evaluating and then comparing the predictive models. Cross-validation approach is a machine learning technique employed to assess and compare the statistical models by partitioning or segmenting the dataset into two portions called as training and test datasets [13]. The training dataset segment of the divided subset is used to learn the model and the remaining data is used to validate the model accuracy. In K-fold cross-validation technique, the model building dataset is first partitioned into K equal (or roughly equal) sized partitions called as the folds [13]. K-1 folds are used for training purpose and the rest 1 fold is used for testing for the final goal of creating each of the K models. The advantage of K-fold-cross-validation lies in its ability to utilize a single dataset for both training and testing and averaging the results across multiple partitions by removing bias. In our study, we apply 10-fold cross-validation for model building and comparison. We evaluate the performance of various models using two different performance parameters such as Accuracy (%) and area under curve (AUC). We conduct statistical tests to identify the best performing change-proneness prediction model.

6.1 Model Building Results

We apply 8 different classification algorithms and 2 different ensemble techniques resulting in 10 different predictive model building approaches. We evaluate the performance of these models using accuracy and AUC metrics. In our study, predicting change-proneness of classes is a binary classification problem and both accuracy and AUC are common evaluation metrics for such problems. Table 7 shows the obtained performance values for set of source code metrics using different classification techniques. From Table 7, it can be inferred that :

- In most of the cases, the model developed by considering selected set of metrics using feature selection techniques as input obtained better performance i.e., high vales of accuracy and AUC for predicting change-proneness as compared to a model developed using all metrics.
- Extreme Learning Machine with polynomial kernel function (ELM-PLY) yields better results when compared to other classification.
- Majority Voting (MV) ensemble method outperformed as compared to all other classifier except ELM-PLY kernel.
- Among different kernel function, polynomial kernel in ELM and RBF kernel in SVM yields better results compared to other kernel functions.

6.2 Comparison of Results

In this section, boxplot analysis has been employed to determine which of the selected set of source code metrics and classification techniques work better for change-proneness prediction.

Source Code Metric Dimensions In this work, we have considered four different set of source code metrics based

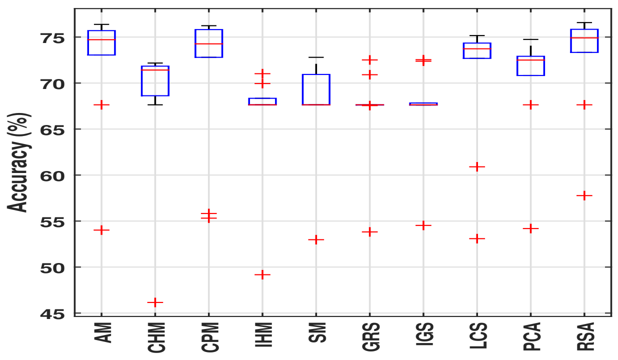
Table 7: Predictive Model Building Performance Evaluation Results

	LOGR	NBC	ELM-LIN	ELM-PLY	ELM-RBF	SVM-LIN	SVM-RBF	SVM-SIG	BTE	MVE
Accuracy										
AM	73.05	54.02	74.46	76.37	74.96	75.59	75.70	67.63	74.46	75.87
CHM	68.62	46.15	71.41	72.17	71.84	70.65	71.78	67.63	71.41	71.84
CPM	73.27	55.82	74.26	76.23	72.79	75.39	75.81	55.31	74.26	76.12
IHM	69.95	49.17	67.63	71.02	67.63	67.63	67.63	68.34	67.63	67.63
SM	71.69	52.97	67.63	70.93	67.63	67.63	67.63	72.79	67.63	67.63
GRS	72.51	53.82	67.61	67.55	67.63	67.63	67.63	70.90	67.61	67.63
IGS	72.37	54.52	67.61	67.83	67.63	67.63	67.63	72.54	67.61	67.63
LCS	72.68	53.09	74.12	75.16	72.91	73.33	74.91	60.90	74.12	74.34
PCA	72.93	54.19	70.82	74.74	72.43	72.57	72.91	67.63	70.82	72.77
RSA	73.33	57.77	74.49	76.57	75.33	75.42	75.84	67.63	74.49	75.87
AUC (Area Under Curve)										
	LOGR	**NBC**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**SVM-LIN**	**SVM-RBF**	**SVM-SIG**	**BTE**	**MVE**
AM	0.70	0.63	0.66	0.69	0.66	0.69	0.67	0.50	0.66	0.67
CHM	0.61	0.58	0.61	0.63	0.62	0.58	0.60	0.50	0.61	0.60
CPM	0.70	0.64	0.64	0.69	0.61	0.70	0.70	0.48	0.64	0.68
IHM	0.64	0.58	0.50	0.58	0.50	0.50	0.50	0.57	0.50	0.50
SM	0.67	0.62	0.50	0.55	0.50	0.50	0.50	0.67	0.50	0.50
GRS	0.68	0.63	0.50	0.50	0.50	0.50	0.50	0.60	0.50	0.50
IGS	0.68	0.64	0.50	0.50	0.50	0.50	0.50	0.66	0.50	0.50
LCS	0.69	0.62	0.65	0.66	0.62	0.63	0.66	0.54	0.65	0.65
PCA	0.69	0.63	0.58	0.66	0.60	0.61	0.62	0.50	0.58	0.61
RSA	0.70	0.66	0.66	0.69	0.67	0.69	0.68	0.50	0.66	0.68

on their measurement properties i.e., cohesion metrics, coupling metrics, size metrics and inheritance metrics. Figure 3 show the box-plot diagrams for accuracy and AUC for each of the set of source code metrics. Box-plot diagrams help to observe performance of all methods based on a single diagram. The line in the middle of each box represents the median value. The model which has high median value is the best model for change-proneness prediction. From Figure 3, it can be inferred, model developed using coupling metrics have high median value of performance parameters as compare to other three set of source code metrics. This results shows that coupling metrics have higher predictive ability as compared to size metrics, cohesion metrics and inheritance metrics.

Feature Selection Techniques: In this paper, five different feature selection techniques are considered to select right set of features for change-proneness prediction. Figure 3 show the box-plot diagrams for accuracy and AUC for each of the set of source code metrics. From Figure 3, we observed that the model developed using selected set of source code metrics using rough set analysis (RSA) have high median value of performance parameters as compare to other. This shows that the model developed using RSA yields better result compared to other approaches. From Figure 3, we also observed that, there exists a small subset of source code software metrics out of total available source code software metrics which are able to predict change-proneness with higher accuracy and reduced value of misclassified errors.

Classification Techniques: In this paper, ten different classification techniques (8 different classification algorithms and 2 different ensemble techniques) are used to develop a change-proneness model by considering source code metrics as input. Figure 4 show the box-plot diagrams for accuracy and AUC for different classification techniques. From Figure 4, we observed that model developed using Extreme Learning Machine with polynomial kernel function (ELM-PLY) have high median value of performance parameters as compare to other classification techniques. From Figure



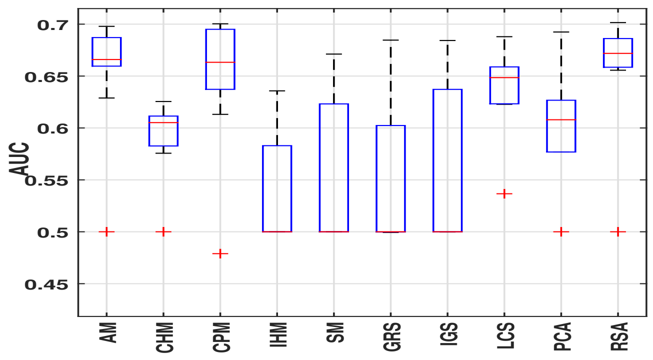


Figure 3: Box Plots Displaying Performance Evaluation Results for Ten Sub-Set of Metrics

4, we also observed that Majority Voting (MV) ensemble method outperformed as compared to all other classifier except ELM-PLY kernel.

6.3 Classifier Technique and Feature Selection Method Interaction :

This section focuses on the affect of classification methods over the performance of feature selection techniques. In this study ten different set of source code metrics and ten different classification techniques have been implemented for investigation. From Table 7, it can be inferred that for each classification method, different set of source code metrics produce better results.

7. THREATS TO VALIDITY

Empirical Software Engineering based experiments and approaches are associated with several potential risks which can affect the validity of experimental findings [27][29][20]. We believe that our work may suffer from the following threats to validity:

Internal Validity: We acquired the program of two different version of Eclipse applications from web-links: [Link 1][6] and [Link 2][7]. Any faulty information or errors not explicitly mentioned in the data and program sources were not considered in our study. We do not make claims or conclusions about the accuracy of data sources as we are not the owners of the data sources, but we believe that the data is collected consistently and correct.

Construct Validity: In the work presented in the paper, developed predictive models for change-proneness detection only forecasts whether a class will change or not, but does not indicate or present any information on the possible number of changes in the class. Nonetheless, we so believe that this issue or limitation is a potential threat to the construct validity of the dependent variable (maintainability prediction) and needs to be systematically eliminated in our follow-up work.

External Validity: We present a case study of an object-oriented systems implemented in Java (Eclipse). We believe that the approach and statistical models designed in our study are likely to be valid for other object-oriented systems as well as programing languages. Further research needs to be carried-out to tailor the approach for other programing paradigms.

8. CONCLUSION

This paper proposes a comparative study of different set source code metrics for change-proneness prediction. The effectiveness of these set of source code metrics are evaluated using eight different learning algorithms such as Logistic Regression (LOGR), Naive Bayes Classifier (NBC), Extreme Learning Machine (ELM) with linear (LIN), polynomial (PLY) and Radial Basis Function (RBF) kernels, Support Vector Machine (SVM) with linear (LIN), RBF and Sigmoid kernel (SIG) and two ensemble techniques such as Best-in-Training (BTE) and Majority Voting (MV). The objective of this study is to investigate the ability of source code metrics to predict change-prone classes. Our main observations are the following:

- Coupling metrics have higher predictive ability as compared to size metrics, cohesion metrics and inheritance metrics.
- From experimental results, it is observed that, there exists a small subset of source code software metrics out of total available source code software metrics which are able to predict change-proneness with higher accuracy and reduced value of misclassified errors.
- From Figure 3, we observed that the model developed using RSA yields better result compared to other approaches.
- From Figure 4, we observed that model developed using Extreme Learning Machine with polynomial kernel function (ELM-PLY) yields better result as compare to other classification techniques. From Figure 4, we also observed that Majority Voting (MV) ensemble method outperformed as compared to all other classifier except ELM-PLY kernel.
- From experiments, it is observed that the performance of the feature selection techniques is varied with the difference classification methods used.

[6]http://bit.ly/2gGpYIO
[7]http://bit.ly/2h5elbe

9. REFERENCES

[1] Basili, V. R., Briand, L. C., and Melo, W. L. A validation of Object-Oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering 22*, 10 (October 1996), 751–761.

[2] Chen, J.-C., and Huang, S.-J. An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software 82*, 6 (2009), 981–992.

[3] Chen, Y., Abraham, A., and Yang, B. Feature selection and classification using flexible neural tree. *Neurocomputing 70*, 1 (2006), 305–313.

[4] Chidamber, S. R., and Kemerer, C. F. A metrics suite for Object-Oriented design. *IEEE Transactions on Software Engineering 20*, 6 (June 1994), 476–493.

[5] Doraisamy, S., Golzari, S., Mohd, N., Sulaiman, M. N., and Udzir, N. I. A study on feature selection and classification techniques for automatic genre classification of traditional malay music. In *ISMIR* (2008), pp. 331–336.

[6] Elish, M. O., Aljamaan, H., and Ahmad, I. Three empirical studies on predicting software maintainability using ensemble methods. *Soft Computing 19*, 9 (2015), 2511–2524.

[7] Forman, G. An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research 3* (2003), 1289–1305.

[8] Furlanello, C., Serafini, M., Merler, S., and Jurman, G. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC bioinformatics 4*, 1 (2003), 1.

[9] Gao, K., Khoshgoftaar, T. M., Wang, H., and Seliya, N. Choosing software metrics for defect prediction: An investigation on feature selection techniques. *Softw. Pract. Exper. 41*, 5 (Apr. 2011), 579–606.

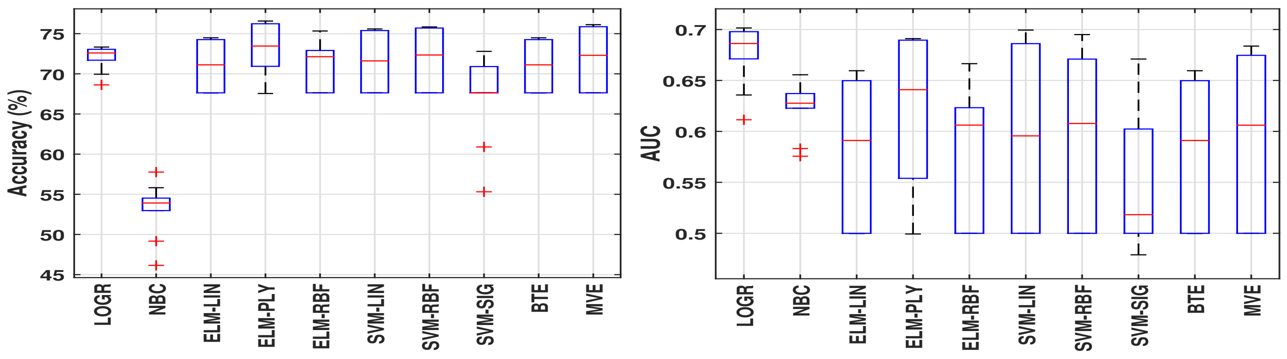


Figure 4: Box Plots Displaying Performance Evaluation Results for Ten Classification Methods

[10] Han, A.-R., Jeon, S.-U., Bae, D.-H., and Hong, J.-E. Measuring behavioral dependency for improving change-proneness prediction in uml-based design models. *Journal of Systems and Software 83*, 2 (2010), 222–234.

[11] Huang, D., and Chow, T. W. Effective feature selection scheme using mutual information. *Neurocomputing 63* (2005), 325–343.

[12] Kabir, M. M., Islam, M. M., and Murase, K. A new wrapper feature selection approach using neural network. *Neurocomputing 73*, 16 (2010), 3273–3283.

[13] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, San Mateo* (1995), pp. 1137–1143.

[14] Kumar, L., Jetley, R., and Sureka, A. Source code metrics for programmable logic controller (plc) ladder diagram (ld) visual programming language. In *Workshop on Emerging Trends in Software Metrics* (2016), WETSoM, ACM, pp. 15–21.

[15] Kumar, L., Rath, S., and Sureka, A. Predicting quality of service (qos) parameters using extreme learning machines with various kernel methods. In *Workshop on Quantitative Approaches to Software Quality (QuASoQ 2016) co-located to (APSEC 2016)* (2016), CEUR.

[16] Lal, S., and Sureka, A. Logopt: Static feature extraction from source code for automated catch block logging prediction. In *Proceedings of the 9th India Software Engineering Conference* (2016), ISEC '16, pp. 151–155.

[17] Li, W., and Henry, S. Maintenance metrics for the Object-Oriented paradigm. In *Proceedings of First International Software Metrics Symposium* (1993), pp. 52–60.

[18] Li, W., and Henry, S. Object-oriented metrics that predict maintainability. *Journal of systems and software 23*, 2 (1993), 111–122.

[19] Lu, H., Zhou, Y., Xu, B., Leung, H., and Chen, L. The ability of object-oriented metrics to predict change-proneness: a meta-analysis. *Empirical software engineering 17*, 3 (2012), 200–242.

[20] Malhotra, R., and Chug, A. Application of group method of data handling model for software maintainability prediction using object oriented systems. *International Journal of System Assurance Engineering and Management 5*, 2 (2014), 165–173.

[21] Malhotra, R., and Jain, A. Fault prediction using statistical and machine learning methods for improving software quality. *Journal of Information Processing Systems*.

[22] Marcus, A., and Poshyvanyk, D. The conceptual cohesion of classes. In *Proceedings of the 21st IEEE International Conference on Software Maintenance* (2005), ICSM '05, pp. 133–142.

[23] Offutt, J., Abdurazik, A., and Schach, S. R. Quantitatively measuring object-oriented couplings. *Software Quality Journal 16*, 4 (2008), 489–512.

[24] Sheldon, F. T., Jerath, K., and Chung, H. Metrics for maintainability of class inheritance hierarchies. *Journal of Software Maintenance and Evolution: Research and Practice 14*, 3 (2002), 147–160.

[25] Thangavel, K., Shen, Q., and Pethalakshmi, A. Application of clustering for feature selection based on rough set theory approach. *AIML Journal 6*, 1 (2006), 19–27.

[26] Xia, X., Lo, D., Correa, D., Sureka, A., and Shihab, E. It takes two to tango: Deleted stack overflow question prediction with text and meta features. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (June 2016), vol. 1, pp. 73–82.

[27] Xu, B., Lo, D., Xia, X., Sureka, A., and Li, S. Efspredictor: Predicting configuration bugs with ensemble feature selection. In *2015 Asia-Pacific Software Engineering Conference (APSEC)* (Dec 2015), pp. 206–213.

[28] Zhao, J., Lu, K., and He, X. Locality sensitive semi-supervised feature selection. *Neurocomputing 71*, 10 (2008), 1842–1849.

[29] Zhou, Y., and Leung, H. Predicting object-oriented software maintainability using multivariate adaptive regression splines. *Journal of Materials Processing Technology 80*, 8 (2007), 1349–1361.