A comparative analysis of evolutionary algorithms for the prediction of software change

Loveleen Kaur
Deptt. of Comp. Science & Engineering,
Thapar Institute of Engineering & Technology,
Patiala, India.
loveleen.kaur[at]thapar.edu

Ashutosh Mishra
Deptt. of Comp. Science & Engineering,
Thapar Institute of Engineering & Technology,
Patiala, India.
ashutosh.mishra[at]thapar.edu

***Abstract*—Change-proneness prediction of software components has become a significant research area wherein the quest for the best classifier still persists. Although numerous statistical and Machine Learning (ML) techniques have been presented and employed in the past literature for an efficient generation of change-proneness prediction models, evolutionary algorithms, on the other hand, remain vastly unexamined and unaddressed for this purpose. Bearing this in mind, this research work targets to probe the potency of six evolutionary algorithms for developing such change prediction models, specifically for source code files. We employ apposite object oriented metrics to construct four software datasets from four consecutive releases of a software project. Furthermore, the prediction capability of the selected evolutionary algorithms is evaluated, ranked and compared against two statistical classifiers using the Wilcoxon signed rank test and Friedman statistical test. On the basis of the results obtained from the experiments conducted in this article, it can be ascertained that the evolutionary algorithms possess a capability for predicting change-prone files with high accuracies, sometimes even higher than the selected statistical classifiers.**

Keywords—Software change; source code change; Evolutionary algorithms; change prediction

I. INTRODUCTION

Software change prediction process consists of employing various design metrics to identify those software components which are expected to undergo modification/change in the consequent versions of the software project. The discovery such components is crucial as these need extra attention during testing and maintenance phases of software development and could possibly act as likely foundations of advancements as well as defects [1, 2]. Apposite distribution of resources to these components aids in enhancing the overall software quality since rigorous authentication activities can be carried out on such components during the prevenient stages of software development to facilitate a timely identification of probable changes and defects [2]. Therefore, development of effective change prediction models is imperative for releasing and maintaining high quality and cost effective software products.

The internal parameters of a predictive model have a very important role in efficiently and effectively training a model and producing accurate results. This is where various optimization strategies and algorithms come into play. Such strategies and algorithms update and calculate appropriate and optimum values of the prediction model's parameters which influence the model's learning process and its output. Amidst the varied set of available optimization and search strategies, the development of Evolutionary Algorithms (EAs) [3] have risen to prominence in the last decade. EAs are a collection of contemporary meta-heuristic algorithms that take inspiration from the manner biological evolution and living beings behave. Techniques like genetic algorithm (GA), genetic programming (GP), evolutionary programming (EP), learning classifier systems (LCS), differential evolution (DE), etc all come under the category of evolutionary computation. These EAs often employ Machine Learning(ML) techniques like statistical methods, support vector machines (SVM) and artificial neural networks (ANN) for the purpose of enhancing the algorithm performance. Machine-Learning (ML) techniques aid the EAs in various algorithmic tasks such as local search, fitness evaluation and selection, population initialization, population reproduction and variation, and algorithm adaptation.

Although a number of studies[1, 2, 4-9] have validated ML as well as statistical methods on various datasets (open-source and commercial) for developing efficient software change prediction models, application of Evolutionary Algorithms(EAs) for the same purpose remains vastly unexamined.

Therefore, in this study, we conduct a comparison of the performances of six evolutionary techniques on datasets derived (via nine randomly selected object-oriented metrics) from four successively releases of a Java-based software for predicting version to version source code file change-proneness. Additionally, we examine and conduct the comparison of the results of these evolutionary techniques with two statistical techniques using accuracy as a performance indicator. We employ statistical tests to statistically rank the algorithms based on their predictive performance. Moreover, the technique having the best performance is pairwise compared with the remaining seven techniques in order to assess if it is statistically similar to the other employed techniques. The empirical analysis performed in this research work thus helps in providing answers to Research Questions (RQs) like:
RQ1: Do the employed evolutionary techniques depict satisfactory predictive performance accuracy for change prediction with respect to the JFreeChart datasets?
RQ2: Do the selected evolutionary techniques outperform statistical techniques in predicting the change-prone files?

RQ3: During a pairwise comparative analysis, is the best evolutionary technique similar to other selected evolutionary and statistical techniques?

The research article proceeds as follows: Section 2 consists of the related work, whereas the research preliminaries and research methodology incorporated are discussed in Section 3. Section 4 provides a comprehensive report of the empirical analysis results. Section 5 comprises of the various threats to validity of our work. Lastly, Section 6 provides the conclusions as well as the future work.

II. Related Work

The existing literature consists of a high volume of studies that investigate software change-proneness. As we only consider the Object-Oriented(OO) software metrics in this article, therefore our discussion apropos to the related work is limited to only those research articles that assess prediction algorithms on OO metrics. Neural Nets, Support Vector Machine and Naive Bayes have been examined on datasets created from eight Eclipse plugins and two Hibernate systems for the prediction of change-prone Java interfaces [4]. Authors [5] explored the Bayesian networks and Neural networks to detect whether a code file gets influenced by a specific sort of change in code, using the static source code dependency graph of the Azureus software and 19 plugin projects of Eclipse. Apart from applying the ML techniques, a few of the authors employ the statistical techniques for estimation of change-prone components. For instance, authors [6] explore the associations between 62 OO metrics and change-prone statistic with 102 Java-based software systems via the statistical random effect model. Standard logistic regression models based on maximum likelihood estimation were also examined for predicting change-prone classes with the aid of coupling metrics, from one software release to the next [7].

We also found a few studies related to change prediction where an empirical comparison between the ML and the statistical techniques was performed. Authors in [8] assessed and compared the performances of ML techniques with the statistical method with datasets created from two versions each of Art-of-Illusion and SweetHome-3D software; while authors in [2] employed two versions each of Frinika, FreeMind and OrDrumbox, to analyze the performance of Logistic Regression(LR), Random Forest, Bagging and Multi Layer Perceptron. It was reported by both the articles that the most of the ML techniques significantly outperform the statistical LR technique. Apart from the use of ML and statistical algorithms for change-proneness prediction, few search-based and hybridized methodologies have also been employed recently for predicting change at a class level in six Android application packages [9].

The study conducted in this research work is unlike existing previous works as it analyses, evaluates and compares six evolutionary techniques and two statistical methods for the creation of change prediction models. Despite the fact that there has been the construction of certain change prediction models in the previous research, these models merely evaluate the ability of common machine learning techniques and statistical methods. Also, the analysis for change-proneness has been usually performed at a class level, such as in [9]. The analysis performed in this research work centers on predicting the version to version change-proneness of files, Java files to be specific which could be a class or an interface. Moreover, we perform statistical tests, such as the Friedman's test and Wilcoxon signed rank test, in our study to rank the prediction models according to their performance and to assess if the difference or disparity in the performance of the selected prediction techniques holds statistical significance or not. Such tests could assist the software development team in a competent selection of a prediction methodology amidst an extensive range of existing methodologies.

III. Research Preliminaries & The Research Methodology

Four datasets were generated from four successive releases of the JFreeChart (jfree.org/jfreechart) software, for the purpose of constructing the Java file change prediction models. The JFreeChart software is a free (LGPL) chart library for the Java platform. Numerical values of nine different OO metrics were gathered apropos to every Java file present in each of the four selected JFreeChart versions with the aid of three static code analysis tools, Understand for Java (scitools.com), Stan4J (stan4j.com) and JHawk 6.1.3 (virtualmachinery.com/jhawkprod), out of which the last two are open source. Some of these metrics have been extensively utilized for predicting the change-proneness of classes in the existing literature. However, our analysis also includes metrics like Instability and Maintainability Index which have not been analyzed in any of the previous research works in regard to change-proneness prediction. Brief description of the nine selected metrics has been in Table I. Pursuant to our context of application, these nine software metrics are selected as the independent variables.

The dichotomous variable of change statistic is the dependent variable in our study, which is to be predicted through the independent variables. AntiCut&Paste (anticutandpaste.com), a clone detection tool, was employed to calculate the change statistic of specific Java files contained in each of the JFreeChart releases. Complete source code corresponding to two successive JFreeChart releases was supplied as an input to the tool and those Java files which were detected to be common to the two versions were returned as output. On the basis of this output, a dichotomous variable of a 'Yes' or 'No' was evaluated as the change statistic and was assigned to each JFreeChart version's file with 'Yes' signifying that the file was used with some change into next version and 'No' signifying that the file was used without any modification. The specifics of the four selected JFreeChart releases have been provided in Table II. The research methodology incorporated is explained in the following sub-sections:

TABLE I. METRICS SELECTED FOR CHANGE PREDICTION

Metric and Source	**Description**
Total Cyclomatic Complexity(TCC) [10]	This metric calculates the number of linearly independent paths through a given program code. Understand measures TCC by counting the keywords for decision points and then adding 1.
Cumulative Halstead Length(CHL) [11]	The Cumulative Halstead length (CHL) of a file is the addition of the number of operators(OP) and the number of operands(OD) it contains, i.e. OP+OD.
Cumulative Halstead Volume(CHV) [11]	The Cumulative Halstead Volume (CHV) of a file pertains to the total information that the reader of the source code has to grasp in order to comprehend its implication and is given as = CHL * $\log_2$(VOC); where VOC is Halstead Vocabulary and is evaluated as the sum of the number of distinct operators(UOP) and the number of distinct operands(UOD).
Cumulative Halstead Effort(CHE)[11]	The Cumulative Halstead Effort (CHE) of a file is the total mental effort needed to refabricate the code under consideration and is given as = CHV* DIF; where DIF is Halstead Difficulty and is calculated as (UOP/2) * (OD/UOD).
Cumulative Halstead Bugs(CHB) [11]	The Cumulative Halstead Bugs (CHB) assesses the number of bugs that hold a possibility of being found in a given file and is calculated as = CHV/3000.
Maintainability Index (MI) [12]	The maintainability index (MI) determines the ease of maintenance of a specific body of code and is calculated as : MI = 171 - 3.42ln(aveE) - 0.23aveV(g') - 16.2ln(aveLOC); Where aveE indicates the average Halstead Effort of the file, aveV(g') indicates the average extended cyclomatic complexity of the file and aveLOC is the average numbers of code lines per module in that file.
Afferent Coupling(AC)[13]	The Afferent Coupling metric determines the number of interfaces and classes from other packages which are dependent on class/classes in the examined file.
Efferent Coupling(EC) [13]	Efferent Coupling is identified as the outgoing dependencies or the number of types in a class belonging to a file which are dependent on types of other classes.
Instability [13]	Instability is gauged by measuring the effort to modify a file without affecting other files in the software and is calculated by comparing the incoming and outgoing file dependencies according to the formula: Instability I = EC / (AC + EC).

TABLE II. SPECIFICS OF THE JFREECHART RELEASES EMPLOYED

JFreeChart version	**Total size(in LOC)**	**Total number of Java files**	**Number of Java files used in the next version**	**Number of Java files used into next version with change**
JFC 0.6.0	16,445	86	86	19
JFC 0.7.0	20,723	105	102	60
JFC 0.7.1	24,171	128	122	28
JFC 0.7.2	24,961	130	130	18

A. *Evolutionary Algorithms*

This section presents a concise explanation of the six evolutionary algorithms employed in this research for change-proneness prediction. We employ four algorithms belonging to the Evolutionary Fuzzy Rule Learning approach and two algorithms from the Evolutionary Neural Network category, the brief descriptions of which are provided in Table III.

In addition to the employed evolutionary techniques, our research work also includes the evaluation of two commonly employed statistical classification techniques of *Linear Discriminant analysis* (LDA) and *Binary Logistic regression* (BLR). Although LDA is more sensitive to outliers, it is observed to obtain higher classification accuracy than BLR when all its necessary requirements are met. (For more information, please refer to [19]).

B. *Performance evaluation measure*

The accuracy obtained during the training and testing phases of model validation is employed to evaluate the potential of the selected algorithms for predicting version to version source code file change-proneness and is calculated as the percentage of the number of Java files that have been predicted accurately to the total number of Java files in that version.

C. *Application of statistical tests for comparison*

This empirical work employs the non-parametric Friedman's statistical test [9], for the purpose of allocating mean ranks to every selected technique on the basis of its prediction performance in regard to the four JFreeChart data sets. The technique with the highest mean rank is adjudged as the best technique. Additionally, this work also conducts the pairwise performance analysis of the best technique with the rest of the selected prediction techniques via the Wilcoxon signed rank test [9].

IV. EMPIRICAL ANALYSIS RESULTS AND DISCUSSION

The evaluation results of the models generated via six evolutionary algorithms and two statistical classification techniques to predict the version to version change-proneness of Java files on the four selected JFreeChart versions have been provided and discussed in this section. The predictor variables incorporated to develop the models were the software metrics stated in Table I. These results were predicted using the Java-based KEEL tool (accessible at www.keel.es), developed by a unit of Spanish research institutes. The default settings of KEEL were used were used to construct the models corresponding to the statistical techniques of LDA and BLR and for GFS-LB and GFS-AB. For NNEP and GANN, the maximum number of generations were brought down to 10 and 100 backpropagation cycles were employed to generate the models corresponding to GANN. All the evolutionary algorithms were run for 30 iterations to produce the results for the purpose of obtaining a rigorous statistical evidence with respect to their performance.

TABLE III. DETAILS OF THE SIX EVOLUTIONARY TECHNIQUES EMPLOYED IN THIS STUDY

Evolutionary approach	**Algorithm employed**	**Description**
Evolutionary Fuzzy Rule Learning	GFS-GP: Fuzzy Learning based on Genetic Programming	In GFS-GP [14], the working of the Fuzzy Learning approach is optimized using search techniques like GP which uses a linear or tree genotype coding of the rule bases.
	GFS-SP: Fuzzy Learning based on Genetic Programming Grammar Operators and Simulated Annealing	On the other hand, the GFS-SP [14] are simulated annealing based techniques that employ the phenotype- genotype type to encode the fuzzy rule bases. A macromutation, derived from tree-shaped genotype GA, is used to replace the adjacency operator in SA. Results of GFS-SP are often observed to be comparable to GFS-GP, when it comes to the efficiency of the classifiers but learning process in GFS-SP requires less memory than in GFS-GP.
	GFS-AB: Fuzzy AdaBoost	There are two benefits of employing boosting methods for fuzzy classifier learning [15]: (1) the learning is quick; and (2) the size of the rule base is between the lowest values attainable with other genetic fuzzy classifiers. Adaboost is a specialised version of the backfitting algorithm that produces reiterative estimates to a maximum likelihood estimation over a set of additive models.
	GFS-LB: Fuzzy LogitBoost	The Logitboost algorithm [16] validates the exponential bound presented in the Adaboost algorithm. This exponential bound is used to approximate the objective function that arises when a Generalized Additive Linear Model is employed for fitting a classification problem post the application of a logistic transform.
Evolutionary Neural Network	NNEP: Neural Network Evolutionary Programming	NNEP [17] is a unique variant of feed-forward neural network, known as product-unit neural networks in which the model insinuated changes the network's weights and the structure with the aid of an evolutionary algorithm. This change in the network structure partly eases the difficulty involved in not knowing the best network structure for a given problem in advance.
	GANN: Genetic Algorithm with Neural Network	GANN [18] is the modified hybridized genetic algorithm (GA) and artificial neural network (ANN) method which work in unison to discover the most optimal feature set that is capable of efficiently classifying the data. The GA locates the ANN with the most optimum feature set that will accurately segregate the classes, whereas the fitness values of GA are optimized by ANN.

Additionally, the 10-fold cross validation technique [2] has been used to develop the models and the model results have been appraised by comparing their Accuracy values on the four JFreeChart datasets incorporated in this analysis.

RQ1: Do the employed evolutionary techniques depict satisfactory predictive performance accuracy for change prediction with respect to the JFreeChart datasets?

Table IV reports the version-wise accuracy results of 10-fold cross validation for models generated via the six evolutionary techniques for the testing and training phases using the four JFreeChart data sets. The results of the prediction models generated by means of the evolutionary techniques report accuracy values above 70% in practically all of the cases when the four JFreeChart data set versions were used, for both the training and testing phases. Table IV also shows the average accuracy values scored for each of the version datasets when the selected evolutionary techniques were evaluated. It can be perceived that the average accuracy values on the four data sets were greater than 82.8% for the training phase and higher than 73.7% for the testing phase, which shows the effectiveness of selected evolutionary techniques for creating file change prediction models, thus concluding that developers can employ the evolutionary techniques to predict change-prone Java files from version to version.

TABLE IV. ACCURACY RESULTS OBTAINED BY THE SELECTED EAS

Selected prediction techniques	**JFC 0.6.0**		**JFC 0.7.0**		**JFC 0.7.1**		**JFC 0.7.2**	
	Train	***Test***	***Train***	***Test***	***Train***	***Test***	***Train***	***Test***
NNEP	87.9	77.9	82.5	74.5	78.4	77.1	89.1	86.2
GANN	89.4	80.2	84.3	77.5	78.4	77.1	84.0	76.9
GFS-GP	84.9	80.2	72.7	65.7	81.4	78.8	88.4	84.6
GFS-SP	85.7	80.2	73.9	71.6	79.3	75.4	87.8	87.7
GFS-LB	99.5	77.9	98.7	78.4	93.8	76.2	99.1	86.9
GFS-ADB	95.2	80.2	84.8	74.5	86.3	75.4	93.3	86.2
Average accuracy	90.4	79.4	82.8	73.7	82.9	76.7	90.3	84.8

RQ2: Do the selected evolutionary techniques outperform statistical techniques in predicting the change-prone files?

For the purpose of establishing the competency of the evolutionary techniques against other commonly used techniques for software change prediction, we evaluated two statistical techniques of LDA and BLR, the accuracy results of which were generated using the same JFreeChart version datasets and have been reported in Table V. Again, the results have been gathered after applying 10-fold cross validation. As observed from the results depicted in Table V, there is a disparity (howbeit slight) in the predictive ability of the models generated via the selected (evolutionary and statistical) algorithms.

TABLE V. ACCURACY RESULTS OBTAINED BY THE STATISTICAL CLASSIFICATION TECHNIQUES

Selected statistical techniques	JFC 0.6.0		JFC 0.7.0		JFC 0.7.1		JFC 0.7.2	
	Train	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*
LR	90.9	84.9	82.9	73.5	82.6	79.0	88.5	83.9
LDA	91.5	88.4	77.8	69.6	76.2	76.2	91.6	88.5

Therefore, to know which of the selected evolutionary techniques performs the best for software change prediction overall (testing and training) and to statically assess if the evolutionary techniques outperform the two statistical classification techniques, we carry out the Friedman rank test [9] as reported in subsection C of Section 3.

Table VI states the mean rank obtained by each of the selected techniques post the application of the Friedman test, wherein the worst performer is the model having the least mean rank. The test was based on the training and testing Accuracy results scored by the eight techniques on the selected JFreeChart versions. As per the results reported in Table VI, the best performing technique apropos to file change-proneness on all the four data sets is the evolutionary technique of GFS-LB having a mean rank of 6.44. Another evolutionary technique of GFS-ABD with a mean rank of 5.50 comes next. The statistical techniques of BLR and LDA follow next, with mean ranks of 5.13 and 4.63 respectively, outperforming other four evolutionary techniques. The evolutionary technique of GFS-GP is selected as the worst performer among the eight prediction techniques having the least mean rank of 3.25.

TABLE VI. FRIEDMAN STATISTICAL TEST RESULTS

Algorithm	Mean Rank
NNEP	3.94
GANN	3.94
GFS-GP	3.19
GFS-SP	3.25
GFS-LB	*6.44*
GFS-ADB	5.50
BLR	5.13
LDA	4.63

The Friedman statistical value with degree of freedom seven was calculated to be 12.399. However, a p-value of 0.088 was obtained which displays that the test results acquired are not true at a confidence interval of 95%. Consequently, the null hypothesis of Friedman test essaying that there exists a statistical similarity in the prediction performance of the eight selected techniques is accepted.

RQ3: During a pairwise comparative analysis, is the best evolutionary technique similar to other selected evolutionary and statistical techniques?

As seen in the Friedman statistical test results, the eight techniques (six evolutionary techniques and two statistical techniques) selected for evaluation are not statistically significantly diverse in their performance behavior. Therefore, we perform a pairwise comparison between the best adjudged evolutionary technique, namely the GFS-LB and every other employed prediction technique in this article, in order to detect if the best performing technique is statistically significantly different from the other employed prediction techniques or not.

We use a two-tailed Wilcoxon rank test [9] to perform the pairwise comparison as shown in Table VII. The numerical values corresponding to Z-score for pairwise test and significance values between GFS-LB approach, that has been ranked the most accurate, and other evolutionary and statistical algorithms incorporated in this article were gathered. Results of pairwise Wilcoxon test specify that GFS-LB is statistically significantly diverse only from the NNEP and GFS-ADB techniques and is not significantly different from the remaining four evolutionary and two statistical techniques.

TABLE VII. RESULTS OF 2-TAILED WILCOXON TEST ON TESTING AND TRAINING ACCURACY VALUES OF SELECTED TECHNIQUES.

Modelling Method Pairs	Z score	Asymp. Sig.
GFS-LB & NNEP	**-2.028**	**0.043**
GFS-LB & GANN	-1.960	0.050
GFS-LB & GFS-GP	-1.893	0.058
GFS-LB & GFS-SP	-1.960	0.050
GFS-LB & BLR	-1.820	0.069
GFS-LB & LDA	-1.400	0.161
GFS-LB & GFS-ADB	**-2.100**	**0.036**

V. THREATS TO VALIDITY

Although measures have been taken to elude any possible errors in collecting, filtering and assessing the data throughout the progression of this empirical work, possible threats to the validity of this analysis, which are common to other pragmatic analyses, could still persist. Mature clone detection and source code analysis tools were employed in this work to quantify the OO metrics and the change statistic. Granting we provide no testimony apropos to the exactitude of such software, it is assumed that these software collect data with reliability as they are in fact being used effectually in practicality[20] consequently significantly diminishing the scope of the threat to *construct validity* in our experimental analysis. Additionally, the *internal validity* threat is existent in this study as we do not estimate the causal significance of the selected independent variables on the change-proneness of Java files. Furthermore, the main peril to the *external validity* of our experimentations is that the inferences drawn by us may possibly not generalize to a novel research setting or a related sample and can be restricted to the inspected software project i.e. the selected JFreeChart software releases. With the intention of ascertaining the generalizability of the conclusions drawn in this work, the predictive efficacy of the employed algorithms on related datasets developed by means of other programming languages must be estimated to remove the risks to external validity.

VI. Conclusion and Future Work

This paper comprises of an empirical analysis of six Evolutionary Algorithms (EAs) for predicting version to version change-proneness using software metrics as independent variables. To the best of the authors' knowledge, there is no research work, as of now, which performs the comparative pragmatic analysis of EAs for predicting the change prediction of Java files from version to version. Moreover, the prognostic effectiveness of the employed EAs is also compared against two statistical classifiers. We construct datasets from four successive releases of the JFreeChart software and judge the capability of the models using Accuracy as a performance measure.

The key results of this analysis are summarised as follows:

- The results of the analysis conducted depict suitable predictive capability of the selected EAs (accuracy values above 70% in almost all of the cases with the four JFreeChart data sets, for both the training and testing phases), almost comparable to the statistical techniques under consideration in order to develop competent version to version source code change prediction models.
- The results from the Friedman test depict that the best performing technique in regard to change-proneness of files on all the four data sets is the evolutionary technique of GFS-LB, followed by another EA of GFS-ABD. We also found that these two evolutionary techniques perform significantly higher than the selected statistical techniques.
- The pairwise Wilcoxon test results specify that GFS-LB is statistically significantly diverse only from the NNEP and GFS-ADB and is not significantly different from the remaining four selected evolutionary and two statistical techniques.

Overall, from the accuracy results it is observed that the EAs are viable for the prediction of software change and the software developers can effectively utilize these to successfully strategize the sharing of resources and for the development of high quality software. For the future work, we aim to conduct a comparative examination of the performance of the various ML techniques with the employed evolutionary techniques in order to further testify its dominance against prevailing techniques of software change prediction.

References

[1] S. Eski and F.Buzluca, "An Empirical Study on Object-Oriented Metrics and Software Evolution in order to Reduce Testing Costs by Predicting Change-Prone Classes," International Conference on Software Testing, Verification and Validation Workshops , 2011, pp. 566-571.

[2] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and change proneness", International Journal of Machine Learning and Cybernetics, vol. 4, no. 4, 2012 pp. 273-286.

[3] E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, ser. Natural Computing. Berlin, Germany: Springer-Verlag, 2003

[4] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," In ICSM, 2011, pp. 303– 312.

[5] E. Giger, M. Pinzger, and H. C. Gall, "Can We Predict Types of Code Changes? An Empirical Analysis," in Proceedings of the 9th Working Conference on Mining Software Repositories, 2012, pp. 217–226.

[6] H. Lu, Y. Zhou, B. Xu, H. Leung, and L. Chen, "The ability of object-oriented metrics to predict change-proneness: a meta-analysis," Empirical Software Engineering, vol. 17, 2012, pp. 200–242.

[7] M.O. Elish and A. A. Al-Zouri, "Effectiveness of Coupling Metrics in Identifying Change-Prone Object-Oriented Classes." In: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

[8] R. Malhotra and R. Jangra, "Prediction & Assessment of Change Prone Classes Using Statistical & Machine Learning Techniques," Journal of Information Processing Systems, vol. 13, no. 4, 2017.

[9] R. Malhotra and M. Khanna, "An exploratory study for software change prediction in object-oriented systems using hybridized techniques," Automated Software Engineering, vol. 24, no. 3, 2017, pp. 673-717.

[10] T.J. McCabe, "A complexity measure," IEEE Transactions on software Engineering, vol. 4, 1976, pp. 308-320.

[11] H.M. Halstead, "Advances in software science," In Advances in Computers, vol. 18, 1979, pp. 119-172.

[12] P. Oman, and J. Hagemeister, "Metrics for assessing a software system's maintainability," In Proc. of the conference on Software Maintenance, IEEE, 1992, pp. 337-344.

[13] R.C. Martin, Agile software development: principles, patterns, and practices. Prentice Hall, 2002.

[14] L.Sánchez, I. Couso, and J.A. Corrales, "Combining GP Operators With SA Search To Evolve Fuzzy Rule Based Classifiers," Information Sciences, vol. 136, no. 1, 2001, pp. 175-192

[15] D. Jesus, M. José, F. Hoffmann, L. Junco Navascués, and L. Sánchez, "Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms," IEEE Transactions on Fuzzy Systems, vol.12, no.3, 2004, pp. 296-308.

[16] J. Otero and L. Sánchez, "Induction of descriptive fuzzy classifiers with the Logitboost algorithm" , Soft Computing, vol. 10, no. 9, 2006, pp.825-835.

[17] F. J. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and A. C. Martínez-Estudillo, "Evolutionary product-unit neural networks classifiers," Neurocomputing, vol. 72, no. 1–2, 2008, pp. 548–561.

[18] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," In Proc. 3rd Int. Conf. Genetic Algorithms Their Applications, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 379–384.

[19] M. Pohar, M. Blas and S. Turk, "Comparison of Logistic Regression and Linear Discriminant Analysis: a simulation study," Metodoloski Zvezki, vol. 1, 2004, pp. 143-161.

[20] D. P. Mesquita , L.S. Rocha, J.P.P. Gomes, and A.R.R. Neto, "Classification with reject option for software defect prediction," Applied Soft Computing, vol. 49, 2016, pp. 1085-1110.