Comparative Study of Feature Reduction Techniques in Software Change Prediction

Ruchika Malhotra
Department of Software Engineering
Delhi Technological University
Bawana Road, Delhi, India
ruchikamalhotra@dtu.ac.in

Ritvik Kapoor
Department of Software Engineering
Delhi Technological University
Bawana Road, Delhi, India
ritvikush953@gmail.com

Deepti Aggarwal
Department of Software Engineering
Delhi Technological University
Bawana Road, Delhi, India
deeptiagg16@gmail.com

Priya Garg
Department of Software Engineering
Delhi Technological University
Bawana Road, Delhi, India
gargpriya1304@gmail.com

Abstract— **Software change prediction (SCP) is the process of identifying change-prone software classes using various structural and quality metrics by developing predictive techniques. The previous studies done in this field strongly confer the correlation between the quality of metrics and the performance of such SCP models. Past SCP studies have also applied different feature reduction (FR) techniques to address issues of high dimensionality, feature irrelevance, and feature repetition. Due to the vast variety of metric suites and FR techniques applied in SCP, there is a need to analyze and compare them. It will help in identifying the most crucial features and the most effective FR techniques. So, in this research, we conduct experiments to compare and contrast 60 Object-Oriented plus 26 Graph-based metrics and 11 state-of-the-art FR techniques previously employed for SCP over a range of 6 Java projects and 3 diverse classifiers. The AUC-ROC measures and statistical tests over experimental SCP models indicate that FR techniques are effective in SCP. Also, there exist significant differences in the performance of the different FR techniques. Furthermore, from this extensive experimentation, we were able to identify a set of the most effective FR techniques and the most crucial metrics which can be used to build effective SCP models.**

Keywords—Software change prediction, Feature reduction, Object-Oriented metrics, Graph-based metrics, Machine Learning

I. INTRODUCTION

Iterative software development, functionality updates, and extensive software maintenance support are becoming a standard in today's dynamic software industry. Accommodating customer's demands, code refactoring, changes in technology, and fault correction are some of the reasons which contribute to code changes leading to newer versions and releases of the software [1]. These change-related activities can be costly and time-consuming for large and long-running software projects. Hence, industries are nowadays employing Software Change Prediction (SCP) to identify the classes that are most prone to changes in the subsequent releases of the software [2, 3, 4]. Identification of such change-prone classes aids in careful planning, easy maintenance, and proper management of the software projects. It helps the project managers to focus their limited resources on such change-prone classes in advance. It ultimately leads to the development of high-quality software that is maintained and updated in the scheduled time [2].

Recently, various Machine Learning (ML) techniques have been used to predict the change-proneness of the classes [2-5]. These techniques use a variety of independent features for prediction like OO metrics [5], class dependency graph-based metrics [6], class evolution-based metrics [7], developer-related factors [8], etc. Many studies often combine different metric suites to obtain higher performance in SCP [9]. However, this increases the dimensionality of the dataset and inadvertently introduces many irrelevant and redundant features [10]. Working with such high-dimensional datasets brings several challenges like high-computation costs and overfitting of models [11].

Many feature reduction (FR) techniques have been developed to counter these issues. These techniques reduce the dimensionality of data by eliminating the features which are not correlated or useful in finding the outcome variable. Past SCP studies have applied filter-based ranking, wrapper-based subset selection, neural-network based, metaheuristic and extraction-based methods to find effective feature representations [2, 4, 12-16].

Since many diverse metrics have been used and a large variety of FR techniques are available to select the best among these metrics, it is crucial to analyse and compare them. This observation motivates us to conduct an extensive comparative study of 11 FR techniques in SCP. We use relevant Object-Oriented (OO) metrics and Graph-based metrics as independent features since they are the 2 most popularly used metric suites in SCP [2-4, 13, 17]. The experiments are conducted on 6 new datasets that we have collected through proprietary tools. AUC-ROC is used along with statistical tests for evaluation. The objective of our study is three-fold. Firstly, this study will help in the identification of the most important metrics in SCP. Secondly, this study will help in the identification of the most effective FR techniques for SCP. Finally, we can observe how the utility of these metrics and FR techniques varies with changes in the classification techniques.

Broadly, our study answers the following research questions (RQs) -

1. What is the significance of using FR techniques in SCP?
2. Which FR technique is most effective in SCP? Does the type of classifier influence the choice of FR technique?
3. Which features are most important in SCP?

The major contributions of this study are -

1. The study evaluates the performance of 11 FR techniques for SCP. These 11 techniques belong to

diverse categories of FR techniques like filter-based, wrapper-based, extraction-based, neural-network based, and metaheuristic techniques. To the best of the author's knowledge, this is the first extensive study that compares FR techniques in SCP.

2. The study is conducted on 6 new Java datasets that we have collected using open-source and proprietary tools. These datasets have OO metrics and graph-based metrics as independent features. The datasets are diverse in terms of domain, size, source of origin, and class-ratio. Further, the use of 3 classification techniques - Support Vector Machine (SVM), Naive Bayes (NB), and Decision Tree (DTree) with all combinations of 11 FR techniques and 6 datasets increases the generalizability of our results.
3. To establish the basic performance of the classifiers we use AUC-ROC as the performance measure since it is the recommended choice for the imbalanced SCP datasets. To increase the significance of our findings, the Friedman test along with the post-hoc double Scott-Knott test is used for further validation of performance of the FR techniques.

The rest of the paper is organized into 6 sections. Section II describes the past SCP studies and the various FR used by them. Section III provides details about the 11 FR techniques that are compared in this study. Section IV elaborates the experimental framework which includes the data collection procedure, independent features - OO metrics and graph-based metrics, classification techniques, performance indicators, statistical tests, and the overall experimental setup. Section V presents the experimental findings and answers the RQs mentioned above. Section VI presents the threats to validity. Section VII concludes the study and future work.

II. Literature Review

For doing software-related predictive modelling we first study the software project, process, and product, and then extract the metrics that describe them. Malhotra and Bansal [17] reviewed SCP studies and found that product-based metrics such as CK [10], Lorenz and Kidd [18], MOOD [19], QMOOD [20], etc. have been predominantly used in this field for model training. Malhotra et al. [2] surveyed 38 SCP studies and found that most of them used OO metrics for software change prediction. Catolino et al. [8] proposed a study with the use of developer-related metrics to enhance the prediction performance in SCP by 22% as compared to single-version models. Kagdi and Melatic [7] proposed a study that focuses on the need to combine single-version and evolutionary dependencies based on multiple versions. The study concluded that the combination improves the overall performance of SCP. Yang et al. [21] proposed a different approach for training SCP models by using metrics based on the graphical structure of the software. Bhattacharya et al. [6] reflected the utility of graph-based metrics for software predictive modelling fields like bug severity, maintenance, and defect prediction. Giger et al [22] advocated using a combination of OO and graph-based metrics.

To cover all the aspects of modern software, a large set of features are considered for model training. This increase in the feature set size causes problems related to high data dimensionality like unsatisfactory results of predictive classifiers [23, 24, 25]. These studies suggest the use of FR techniques to eliminate repetitive and irrelevant features.

Blum and Langley [26], Tang et al. [27], Chandrashekar, and Sahin [28] have given invaluable insights surveying and explaining the work in ML on FR techniques. FR techniques can be broadly classified into different groups based on the approach and framework followed by them. Some of them are -filter-based methods, wrapper-based methods, feature extraction-based methods, evolutionary methods, and neural network-based methods. Malhotra et al. [2] conducted a review of primary studies on SCP and found 58% of them were using FR techniques. Chi-Square test [8], CFS [9], wrapper method [14], PSO [15], Genetic Algorithm [16], PCA [23] are some of the techniques that have been previously used in SCP for the specified purpose.

Since a large variety of metrics and FR techniques have been applied in SCP, there is a need to compare them to identify the most effective ones. Many such comparative studies are already present for software related fields like software risk analysis, software cost estimation, key class identification, and software defect prediction. On the contrary, no comparative study providing definitive guidance on FR techniques is present for SCP.

On the lines of works by Shivaji et al. [24] and Xu et al. [25] on the comparison of FR techniques in SDP, we propose a similar study in SCP where we investigate the effects of 11 FR techniques using NB, SVM, and DTree classifiers over 6 Java projects - ant, hibernate, wro4j, tomcat, javaclient, and neuroph.

III. Feature Reduction Techniques

High-dimensionality datasets often create various issues like high computational cost, reduction of the impact of important features in the model, and low prediction performance due to overfitting [11]. FR techniques counter these problems by selecting, removing, or transforming the original features to reduce the dimensionality of the dataset [24, 25]. This study compares 11 FR techniques belonging to different categories. These techniques are listed in Table I. These techniques are explained in the subsections below.

TABLE I. FR Techniques

Category	FR Technique	Abbreviation
Filter-based	Chi-Square	CS
	Gini Index	Gini
	ReliefF	RF
	Fisher Score	FS
Wrapper-based	Forward Learning	Wrap
Extraction-based	Principal Component Analysis	PCA
	Linear Discriminant Analysis	LDA
Metaheuristic	Particle Swarm Optimization	PSO
	Artificial Bee Colonization	ABC
	Firefly Algorithm	FF
Neural-Network based	Auto-Encoders	AE

A. Filter-Based Techniques

Filter-based techniques reduce the number of features by providing a rank to the original features according to some statistical or probability-based measure. The features with ranks higher than the threshold are chosen while the others are eliminated in the reduced dataset. This threshold value is an external variable. In this study, we have used four different feature ranking mechanisms –

1) Chi-Square: CS [12] tests the dependency of features with the dependent variable and selects those with the highest dependence. Higher the CS value for a feature, the more effective it is in the prediction of the dependent variable.

2) Gini Index: Gini [29] is a splitting criterion that is used in DTrees. Gini calculates the entropy of the dependent variable with each feature. Gini is inversely related to the effectiveness of the feature.

3) ReliefF: RF [25] is a supervised multivariate feature selection (FS) technique that assigns a feature score to each feature. A feature is assigned a higher score if its value remains similar across samples of the same class and changes across samples of different classes. Features having higher scores are chosen for model training.

4) Fisher Score: FS [25] is a statistical method for calculating the rank of each feature. A Higher Fisher score implies a higher priority of a feature. FS is calculated as follows:

$$\mathrm{FS} = \frac{n_p(\mu_p + \mu_T)^2 + n_N(\mu_N - \mu_T)^2}{\sigma_T{}^2} \tag{1}$$

where μ_T is the mean of the values in the dataset, σ_T is the standard deviation of the dataset, N signifies negative samples and p signifies positive samples for the original dataset.

B. Wrapper-Based Techniques

Wrapper-based techniques [14] use classification techniques to select the best feature subset by training for different combinations of features. Wrapper-based techniques use forward learning, backward elimination, or a combination of both for searching for the best combination of features. Wrapper-based techniques follow a greedy-search approach where the classification accuracy is the evaluation measure for every feature combination. In this study 3 classification techniques have been used for wrapper-based FS - NB, DTree, and SVM.

C. Extraction-Based Techniques

Extraction-based techniques project the high-dimensional original dataset into a lower-dimensional subspace. This projection is formed by extracting significant structures and relationships from the original features. These transformed features can be used for training the model. Two such techniques have been used in this study for comparison -

1) Principal Component Analysis: PCA [23] represents the dataset in the form of orthogonal vectors called principal components. These principal components are formed by linear combinations of the original features. For a dataset with n number of features, n principal components are created. A set of k principal components having the highest variance are selected as the new features. Here, k is the reduced dimension size.

2) Linear Discriminant Analysis: LDA [30] is a supervised technique that creates new features considering the dependent variable. The objective of LDA is to project the original dataset to a subspace where the distance between similar class samples is minimized while the distance between samples of different classes is maximized.

D. Metaheuristic Techniques

Metaheuristic techniques are based on the real-life evolution strategies adopted by various living organisms. Their use for feature reduction has recently increased due to their ability to find a globally optimum combination of features in less time. For all the described techniques, AUC-ROC was taken as the fitness function. In this study, 3 metaheuristic techniques have been used for comparison -

1) Particle Swarm Optimization: PSO [15] is a metaheuristic computation technique that has been widely applied in feature selection due to its computational efficiency, fast convergence and its ability to find global optimum solutions. PSO consists of a group of particles moving in the solution space of the problem. Every particle's position represents a possible feature subset. A fitness function evaluates the suitability of a solution for the given problem. The particles move to better positions in every iteration based on their personal experience and the overall experience of the population. The process terminates on achieving the desired fitness function value or after a fixed number of iterations.

2) Artificial Bee Colonization: ABC [31] is a swarm-based metaheuristic technique that models the food-search behavior of bee colonies. In ABC, potential solutions to the optimization problem are represented by position of food sources. In feature selection problem, these positions represent a possible feature subset. The fitness function (the nectar amount at a food source position) represents the AUC-ROC value achieved considering the feature subset of that solution. The employed, onlooker, and scout bees perform various iterations to reach the best food source position. The process terminates on achieving the desired fitness function value or after a fixed number of iterations.

3) Firefly Algorithm: FF [32] is a metaheuristic technique based on the flashing behavior of the fireflies. The position of fireflies represents potential solutions to the optimization problem. The intensity of the firefly is the value of the fitness function on that position. On each iteration, fireflies move to better position depending on the bioluminescent attraction between them. The process terminates on achieving the desired fitness function value or after a fixed number of iterations.

E. Neural Network-Based Techniques

AE [30] is an artificial neural network that encodes the original feature set into a compressed and efficient feature set. AE is an unsupervised learner with a bottleneck architecture with input features as the original feature set and output features as the encoded feature set. AE aims to reduce noise, eliminate uncorrelated features, and construct the new feature set as close to the original feature set as possible.

IV. EXPERIMENTAL SETUP

The experimental framework used in the study is depicted in Fig. 1. All the individual components are explained in further subsections.

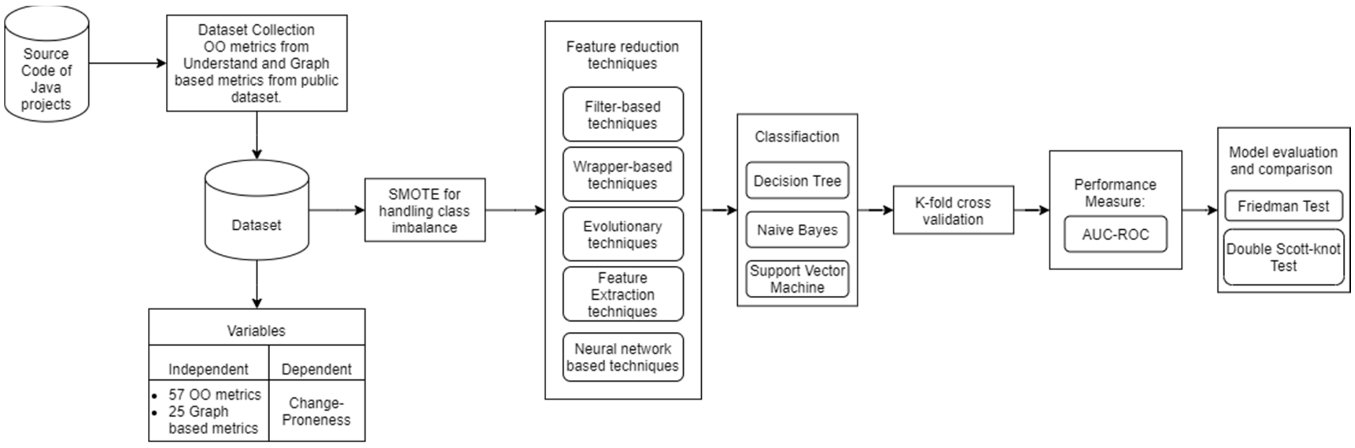

Fig. 1. Experimental Framework

A. *Variables*

1) Independent variables: In this study, two categories of independent variables are used - OO metrics and graph-based metrics. Graph-based metrics are further divided into core metrics and centrality measure metrics. OO metrics [10,18-20] describe the class structure like lines of code, number of methods, number of attributes, cyclomatic complexity, etc. They also represent the relationship between the classes like coupling, cohesion, etc. This study uses 60 handpicked OO metrics[1] collected using the UNDERSTAND tool (http://www.scitools.com/). The graph-based metrics are derived from a dependency graph where each class represents a node and the inter-class relationships like inheritance, containership, etc. form the edges [33]. Core graph metrics calculate features like in-degree, out-degree, weighted connections, etc. of a class, and centrality measures like PageRank, K-Coreness, Betweenness, etc. indicate the importance of a class in the software. This study uses 26 graph-based metrics taken from a publicly available dataset [33]. A total of 86 independent variables have been used in this study for classification. These 86 variables are listed in Table II.

2) Dependent Variables: Change-proneness of a class is the binary dependent variable in this study. There are various types of changes in a class like addition or removal of methods and attributes, modifications in the class declaration, etc. In this study, we model SCP as a binary-classification problem hence all these changes are labeled in the same category i.e. change (1). If no modification has been done in a class between the two subsequent releases, then the sample is labeled as no change (0).

TABLE II. METRICS COMPRISING DATASETS

Category	**Metrics included in the study**
OO metrics	Average Cyclomatic Complexity (F1), Average Modified Cyclomatic Complexity (F2), Average Strict Cyclomatic Complexity (F3), Average Essential Cyclomatic Complexity (F4), Average Number of Lines (F5), Average Number of Blank Lines (F6), Average Number of Lines of Code (F7), Average Number of Lines with Comments (F8), A Base Classes (F9), Coupling Between Objects (F10), Count of Modified Coupling Between Classes (F11), Number of Children(F12), Classes (F13), Class Methods (F14), Class Variables (F15), Executable Unit (F16), Number of Files (F17), Function (F18), Instance Methods (F19), Instance Variables (F20), Local Methods F(21), CountDeclMethodAll (F22), Local Default Visibility Methods F(23), Private Methods (F24), Protected Methods (F25), Public Methods (F26), Inputs (F27), Physical Lines (F28), Blank Lines of Code (F29), Lines of Code (F30), Declarative Lines of Code (F31), Executable Lines of Code (F32), Lines with Comments (F33), Outputs (F34), Paths (F35), Paths Log(x) (F36), Semicolons (F37), Statements (F38), Declarative Statements (F39), Executable Statements (F40), Cyclomatic Complexity (F41), Modified Cyclomatic Complexity (F42), Strict Cyclomatic Complexity (F43), Essential Complexity (F44), Knots (F45), Max Cyclomatic Complexity (F46), Max Modified Cyclomatic Complexity (F47), Max Strict Cyclomatic Complexity (F48), Max Essential Complexity (F49), Max Knots (F50), Depth of Inheritance Tree (F51), Nesting (F52), Minimum Knots (F53), Lack of Cohesion in Methods (F54), Modified Lack of Cohesion in Methods (F55), Comment to Code Ratio (F56), Sum Cyclomatic Complexity (F57), Sum Modified Cyclomatic Complexity (F58), Sum Strict Cyclomatic Complexity (F59), Sum Essential Complexity (F60)
Graph-based metrics	Weighted Incoming Dependencies (F61), Weighted Outgoing Dependencies (F62), Total Weighted Dependencies (F63), PageRank on Directed Graph (F64), PageRank on Unweighted Graph (F65), PageRank on Undirected, Weighted Graph (F66), PageRank on Weighted Graph With Back Recommendations (F67), PageRank on Weighted Graph (F68), Authority Value on Directed Unweighted Graph (F69), Authority Value on Directed, Weighted Graph (F70), Hub Value on Directed Unweighted Graph (F71), Hub Value on Directed Unweighted Graph (f72), Betweenness Value on Undirected Unweighted Graph (F73), Betweenness Value on Undirected Weighted Graph (F74), Core Number on

	Undirected, Unweighted Graph (F75), Core Number on Weighted Graph (F76), Top Classes That Are Incoming Dependencies (F77), Top Classes That Are Outgoing Dependencies (F78), Total Top Class That Are Dependencies (F79), Total Weight of Incoming Dependencies From Top Classes (F80), Total Weight of Outgoing Dependencies From Top Classes (F81), Total Weight of All Dependencies From Top Classes (F82), Betweenness Value on Weighted Graph With Back Recommendation (F83), Incoming Dependencies (F84), Outgoing Dependencies F(85), Total Dependencies (F86).

B. Dataset

This section describes the procedure of dataset collection and the properties of the collected datasets. 6 open-source Java projects are chosen for collecting the datasets [34]. For each project, the source code of two subsequent releases is collected from GitHub (https://github.com) or Sourceforge (https://sourceforge.net). From the source code of both releases, 60 OO metrics[1] for JAVA classes are collected using the UNDERSTAND tool (http://www.scitools.com/). Now the two releases of the project are compared to identify the classes in which there have been some changed or modified classes. Firstly, the classes from both releases are compiled together in the dataset. The classes which are not common to both the releases were classified as changed. For the remaining common classes, the number of lines increased, decreased, and modified from first to the second release are found. For every class, if no lines were added, deleted, or modified it is labeled as no change (0) else change-prone (1). Following the above steps, we get our data with OO metrics and the dependent variable. Now the 26 graph-based metrics available as publicly accessible datasets [33] are combined with the OO metric data collected in previous steps though inner join over the columns containing class names in both datasets This completes our data collection procedure. Finally, we have 6 datasets with 86 independent variables and 1 dependent variable described in Section IV-A. These datasets and their statistics are shown in Table III.

TABLE III. STATISTICS OF DATASETS

Dataset	Release 1	Release 2	Number of classes	% of changed classes	Description
Ant	1.6.0	1.6.1	319	10.3	Java-based build tool of Apache
Tomcat	9.0.3	9.0.4	354	1.97	Open-source implementation of Java Servlet by Apache
Wro4j	1.6.2	1.6.3	169	24.7	Java project that combines web tools
Hibernate	5.2.11	5.2.12	2022	6.77	Object Relational Mapping (ORM) tool
Javaclient	2.2.0.0	2.2.0.1	211	4.24	Appium project on Java
Neuroph	2.1.1	2.2	112	26.5	Java class library and GUI tool

C. Classifiers

In this study, 3 classification techniques have been used for the prediction of change-prone classes. These classifiers have been widely used in the previous comparative studies [24, 25]. Also, these techniques have proven to have good prediction performance and are stable for imbalanced data [35]. These techniques are very diverse from each other in terms of learning strategies and have fast computation efficiency [36]. These classifiers are explained below-

1) Naive Bayes - NB [36] is a statistical probability-based ML technique that predicts the probability of occurrence of a sample in different classes using prior and posterior probabilities and then labels the sample with the class having maximum probability.

2) Support Vector Machine - SVM [36] is a popular supervised ML technique often used in classification and regression problems. SVM develops a hyperplane to divide the sample space such that the distance of the hyperplane with the closest sample is the largest.

3) Decision Tree - DTree [36] constructs a tree where each node represents a condition on a feature and each path from the root to leaf node leads to an output label. While moving from root to leaf various decisions take place according to the features of the samples and the conditions on those features. This is a widely used technique in various problem domains.

D. Performance Measures

In this study, we use AUC-ROC for evaluating the performance of the different techniques. AUC-ROC has been widely used in past studies on SCP [3, 5, 13, 14]. AUC-ROC measures the capability of a model to distinguish between change-prone and no-change classes. It is a summary of the ROC curve which is a plot between True Positive Rate (proportion of change-prone classes that were identified correctly as change-prone) and False Positive Rate (proportion of no-change classes that were identified as change-prone). AUC-ROC values range between 0 to 1 where 1 means that the model is identifying every sample correctly and 0 means that the model is performing poorly and reciprocating the actual class labels. AUC-ROC is a highly effective performance measure especially when the dataset is imbalanced, which is often the cases with datasets in SCP.

E. Statistical Tests

This study uses the non-parametric Friedman test [37] followed by the post-hoc double Scott-Knott test [38] for a consolidated performance comparison of different FR techniques across datasets. The tests are conducted at the significance level (α) of 0.05. To conduct these tests the following steps are followed -

1) Formulation of the null hypothesis (H_0) - H_0 states that all the FR techniques are similar in performance in SCP. In other words, the use of one FR technique over another does not affect the classification performance.

2) Computing Average Friedman Ranking (AFR) - Consider we have f FR techniques and d datasets. For every FR technique, the average rank over d datasets is conducted based on performance measures described in Section IV-D. The formula for calculating AFR of i-th FR technique is –

$$AFR_i = \frac{1}{d}\sum_{j=1}^{d} R_{ij} \quad (2)$$

where R_{ij} is the rank of the i-th technique for the j-th dataset.

3) Computing Friedman test statistic (FTS) - Using the AFR of the FR techniques computed in Step (2), FTS is computed as –

$$FTS = \frac{12d}{f(f+1)}\left(\sum_{i=1}^{f} AFR_i^2 - \frac{f(f+1)^2}{4}\right) \quad (3)$$

4) Accept or Reject H_0 - If the FTS computed in equation (3) is greater than the chi-square distribution value for significance level (α) and f-1 degree of freedom, the null hypothesis H_0 is rejected else H_0 is accepted. if H_0 is rejected, we conduct the post-hoc test according to step (5)

5) Scott-Knott test - It creates distinct clusters of FR techniques using hierarchical clustering. Clusters are created based on a performance metric of FR techniques for a given dataset. Ranks are assigned to techniques based on their cluster.

6) Double Scott-Knott test - It is used for comparison of the techniques across datasets. For each technique, we obtain 6 Ranks using the Scott-Knott test each corresponding to a unique dataset. Now, using dataset wise ranks as an input, we conducted a second Scott-Knott test to obtain overall ranks for an FR technique.

F. Experimental Procedure

The experiments are performed using the datasets described in Table III. Since the datasets are highly-imbalanced, we have used Synthetic-Minority Oversampling (SMOTE) [39] to counter this issue. We used a SMOTE ratio of 0.6 [40].

Next, we have applied the FR techniques given in Table I using the parameters described here. Let p be the number of original features. The filter-based and wrapper-based methods are implemented using the Skfeature library of python. The number of reduced features is a variable in the FR techniques that depends on various factors like dataset distribution, the mathematical formula used by filter methods, etc. Hence, for each technique, we used the brute force technique for variable selection to optimize the percentage of feature reduction. We conducted experiments with fraction values in the range of 0.1 to 0.9 with a difference of 0.1 [41]. From these experiments we found out that 0.5*p is the optimal number of features to be taken for filter-based and wrapper techniques. For wrapper-based techniques, we have used a forward learning search strategy.

PCA and LDA are implemented using the Sci-kit learn python library. The number of components in PCA is chosen as 40 as we want 95% variance to be retained in the new features [42].

The parameters for PSO, ABC, FF, and AE are selected by running the algorithms multiple times on different parameter combinations. The metaheuristic techniques and AE are implemented from scratch in python. PSO is implemented with an initial population size of 30, maximum number of iterations to be 50, c1 and c2 with values 1.496 respectively, and the velocity range to be [-2,2]. ABC is implemented with an initial population size of 30, abandonment limit 100, upper/lower limit as [-1,1], and maximum iterations as 50. FF is implemented using population size 20, α=0.8, β=1, Ɣ=0.7, and maximum iterations as 40.

For AE we use Adadelta optimizer, categorical cross-entropy loss function, and relu activation function. The number of epochs is 50 with a batch size of 100.

The classification techniques used in this study are mentioned in Section IV-C. They are implemented with default parameter settings of Sci-kit learn library, and a 10-fold cross-validation strategy is used for training and evaluation of performance. The performance is measured using the indicators described in Section IV-D. Finally, statistical tests are conducted which are explained in Section IV-E. This entire framework is depicted in Fig. 1.

V. Experimental Results

A. RQ1 - What is the significance of using feature reduction techniques in SCP?

The objective of this RQ is to find the need to apply FR techniques in SCP. Here 11 sets of experiments are conducted considering each of the 11 FR techniques. In addition, a set of experiments is conducted without using FR techniques. Each set of experiments consist of 18 different combinations formed by the 6 datasets and 3 classifiers.

The AUC-ROC results obtained from these experiments are shown as box-plots in Fig. 2, 3, 4. The three box-plots are differentiated based on the three different classifiers (DTree, NB, and SVM) used in the experiments. Each of these box-plots shows the variation of AUC-ROC over different datasets for the 11 FR techniques and the one labelled 'No FR' corresponding to the case where the no FR technique is applied. In the box-plot for the SVM classifier, it is observed that all the FR techniques except the filter-based CS have a higher median AUC-ROC value than the case where no FR technique is used. A similar scenario is observed for the DTree classifier where the scenario of not using any FR gives the second-worst median AUC-ROC after AE. The results of the NB classifier are also against the scenario where we do not use any FR technique as here it gives the third-worst median. To summarize, across all the three classifiers the median AUC-ROC score of the best FR technique is higher than the median of no-FR case. Hence, by analysis of the AUC-ROC performance results, we can say that the application of FR techniques is necessary since there are a lot of redundant and unnecessary features that are not effective in SCP.

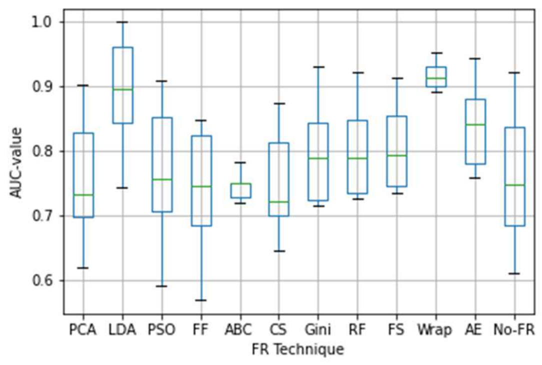

Fig. 2. Box plot of AUC-values over datasets using SVM classifier

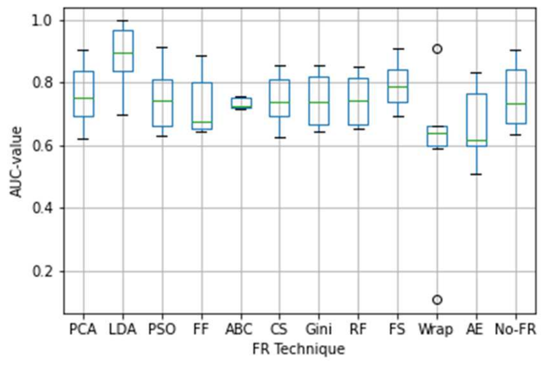

Fig. 3. Box plot of AUC-values over datasets using NB classifier

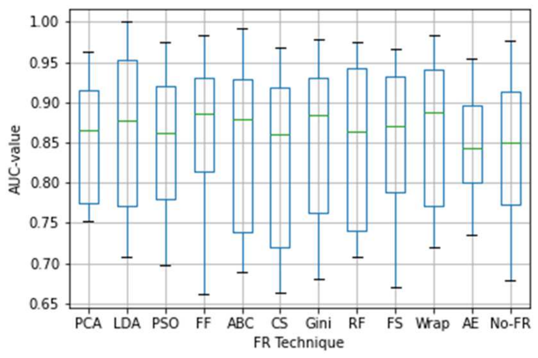

Fig. 4. Box plot of AUC-values over datasets using DTree Classifier

To further validate the inferences drawn from AUC-ROC results, we perform statistical validation. We first applied the Freidman tests where we got the Friedman Statistic Value as 12.295, 22.483, and 25.982 for DTree, NB, and SVM respectively. These values suggest that the initial hypothesis H_0 is rejected for all the three classifiers. This concludes that all 11 FR compared in the study have significant differences in prediction performance. Since we also compared a case where no FR technique was applied, it can be concluded from the rejection of H0 that there is a difference in performance between the scenario where FR techniques are used and where FR techniques are not used. The results of double Scott-Knott tests further justify these observations. The results of the double Scott-Knott test are displayed in figures 5, 6, and 7 using graphs. Here each graph is created based on a classifier used. For each graph, the x-axis hoists the status of the FR technique and the y-axis hoists average rank values. The height of each bar demarcates the average rank of each technique over six datasets. The colour of each bar indicates the distinct cluster to which the corresponding FR technique is assigned. For DTree, the scenario of not using any FR technique is ranked the lowest with an average rank of 8.5. For NB and SVM, the scenario of not using any FR technique is not the worst since there are a few FR techniques like PSO, CS, and FF that perform worse. Table IV summarizes the results of the double Scott-Knott. For each classifier, the cluster division is shown along with the techniques in that cluster and other cluster statistics like mean, median, and standard deviation. It is observed from Table IV that the scenario of not using any FR technique always falls in the cluster of the worst techniques which have the lowest rankings. Since the results of double Scott-Knott tests are independent of the original AUC-ROC values of a technique on the different projects, we can say that the scenario of not using any FR technique is overall undesirable.

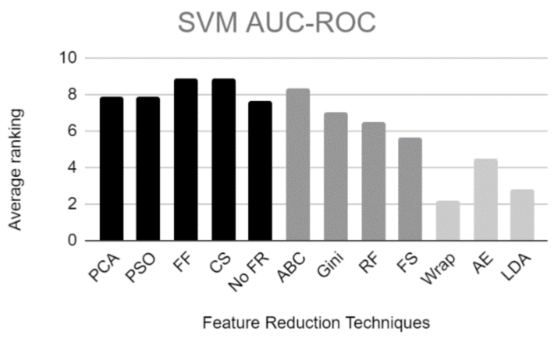

Fig. 5. Double Scott-knot test and Friedman ranking over datasets for SVM classifier

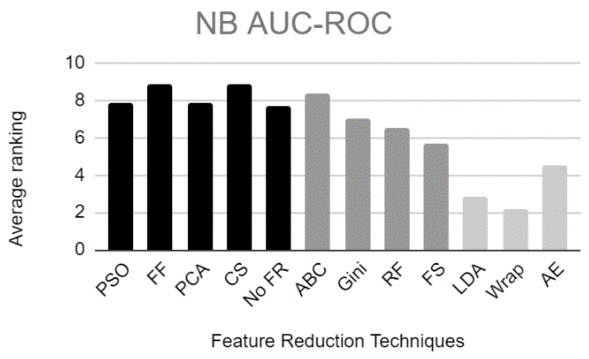

Fig. 6. Double Scott-knot test and Friedman ranking over datasets for NB classifier

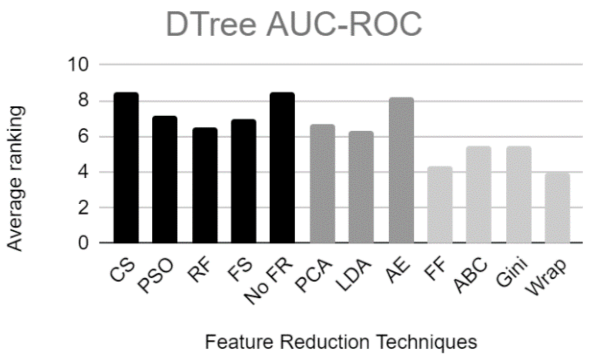

Fig. 7. Double Scott-knot test and Friedman ranking over datasets for DTree Classifier

Conclusion - The experimental results indicate the presence of irrelevant and redundant features that affect the prediction performance of the classifiers. The AUC-ROC values and the statistical tests strongly suggest the necessity of using FR techniques in SCP to select effective features for training predictive models.

TABLE IV. SUMMARY OF DOUBLE SCOTT-KNOTT TEST

DTree					
	Performance Measure	Techniques	Average Rank	Median Rank	Std. Deviation
1	AUC-ROC	FF, ABC, Gini, Wrap	4.8333	4.9165	0.7818
2		PCA, LDA, AE	7.0556	6.6670	0.9767
3		CS,PSO, RF, FS, No FR	7.5333	7.1667	0.9159
NB					
1	AUC-ROC	AE, LDA, Wrap	3.1667	2.8333	1.2018
2		ABC, Gini, RF, FS	6.8749	6.7500	1.1168
3		No FR,CS, PSO, PCA, FF	8.2000	7.8333	0.5821
SVM					
1	AUC-ROC	Wrap, AE, LDA	3.1666	2.8330	1.2019
2		ABC, Gini, RF, FS	6.8758	6.7500	1.1157
3		PSO, FF, CS, PCA, No FR	8.1998	7.8330	0.5820

B. *RQ2- Which feature reduction technique is most effective in SCP? Does the type of classifier influence the choice of feature reduction technique?*

In this RQ, we compare the performance of selected FR techniques in SCP. Also, we study the effect of different classifiers on the performance of the FR techniques.

For SVM, it is observed from Fig. 2 that the least AUC value achieved by Wrap is better than most of the other FR techniques except LDA. The wrapper technique shows good performance as AUC-ROC values over all the datasets lies within the narrow range of 0.89 to 0.93. This shows the superiority of Wrap over other FR techniques. ABC has a lower range of values for these datasets and has the lowest maximum as compared to other techniques. PSO and FF have shown a comparable performance with the highest median for PSO. This observation showed an efficiency of PSO over ABC and FF. All the four filter-based techniques have achieved almost equivalent AUC-ROC range with FS being the most efficient and CS being the least. AE has also performed well due to its mechanism of minimizing errors between input and output features. LDA has shown significantly better results than PCA. This observation indicates that the supervised nature of LDA makes it more efficient than the unsupervised technique PCA.

For NB, it is observed from Fig. 3 that the same trend between LDA and PCA has been followed which shows that this might be true that LDA performs better than PCA independent of the classification technique used in SCP. LDA has performed the best among all the FR techniques in terms of the median of AUC values. For the metaheuristic techniques, it is seen that for PSO and ABC, the median AUC-ROC lies above 0.7 which shows a great predictive efficiency. FF in turn has a comparatively worse median showing the superior performance of PSO and ABC in this case. Filter-based techniques especially CS, Gini, RF have shown equivalent median values. FS has shown the best performance among all the filter-based techniques. Wrap has extremes at very low and high AUC-ROC which shows a change in performance efficiency with datasets.

For DTree, it is observed from Fig. 4 that the minimum AUC value lies above 0.65 which shows that DTree is an effective classification technique. 3 out of 4 filter-based techniques i.e. CS, RF, and FS have achieved almost similar median AUC-ROC but the range of all of these techniques differ. This shows that some filter-based FR techniques are more efficient in selecting better features than the others. AE and FF have comparatively shown less range of values than others. LDA has given the best highest AUC-value as compared to any other techniques. In this case, the performance of all the FR techniques have been comparable unlike NB and SVM. This shows that classifiers widely affect the performance of FR techniques.

The results in Fig. 5, 6 and 7 conclude that some feature techniques are more effective and some are least effective irrespective of the classification technique used. For all three classifiers in Fig. 5, 6, and 7, PSO and No FR have been among the highest ranks, i.e. lowest prediction performance. For NB and SVM, LDA has been among the top performers with rank 2.83, which shows the capability of LDA in efficient feature reduction in SCP. But for DTree, LDA has achieved a rank of 6.33 which lies in the cluster with the second-highest average ranking showing a difference in its performance according to the classifier.

Table IV has shown the clustering of different FR techniques according to the double Scott-Knott test. One important observation is that at least two metaheuristic techniques always lie in the same cluster i.e. FF and ABC or PSO and FF. This shows that metaheuristic techniques tend to perform very similar to each other with less or no significant differences. However, these techniques have always been in the last or second last cluster showing the lower efficiency of metaheuristic techniques in SCP. Wrap technique always lies in the topmost cluster and has achieved the best rank in all the three classifiers. Wrap uses the classification efficiency of the classifier as the evaluation measure to extract the most efficient feature subset. This mechanism helps it in synergizing with corresponding classifiers. The filter-based techniques are divided into various clusters which show that each technique differs from the other in terms of FS.

Conclusion: There are some FR techniques that perform better than other FR techniques. LDA and Wrap have performed very efficiently while PSO and CS have performed worse overall. Filter-based techniques have shown similar results with each other. Metaheuristic techniques have shown good AUC-values but have achieved comparatively lower ranks. It is observed that some techniques have shown consistent results irrespective of the classifier used while the average ranking of various techniques like AE, ABC, FF, etc. differ according to the classifier which shows that classifiers affect the choice of FR techniques.

C. *RQ3- Which features are most important in SCP?*

In this competitive study, we have conducted a total of 204 different experiments to answer questions studying FR techniques in varied aspects. While doing these experiments we generated 48 unique feature rankings of 86 original features. These 48 unique feature rankings were obtained using the 8 FR techniques (CS, Gini, RF, FS, Wrap, PSO, ABC, FF) over 6 unique datasets. AE, LDA, and PCA were not considered because they return new features extracted from the given feature set. Also, the ranks given to features by different FS techniques do not change with a change in classifiers so we have ignored that dimension while answering RQ3.

Now, on the basis of these 48 feature ranks for each feature, we have conducted a Scott -Knott test to form three priority clusters of features. Table V describes the clusters by showing the priority of each cluster and the feature codes of each feature in a cluster. The feature name to feature code mapping is stated in Table II. From studying Scott-Knott test results we have amassed interesting inferences regarding the importance of features.

TABLE V. RANKING OF METRICS USING SCOTT-KNOTT TEST

Cluster No.	Metric Importance	Metrics included in the cluster
1	High	F10, F12, F13, F14, F15, F18, F23, F24, F25, F26, F31, F46, F47, F48, F51, F52, F54, F55, F69, F71, F72, F75, F76, F77, F78, F79, F83, F84, F85, F86
2	Medium	F1, F2, F3, F4, F5, F9, F19, F20, F21, F27, F30, F33, F37, F38, F39, F40, F49, F50, F53, F56, F67, F68, F70
3	Low	F6, F7, F8, F11, F16, F17, F22, F28, F29, F32, F34, F35, F36, F41, F42, F43, F44, F45, F57, F58, F59, F60, F61, F62, F63, F64, F65, F66, F73, F74, F80, F81, F82

Cluster 1 has the highest priority features and it consists of 30 features, 30% of the OO metrics and 46.15% of the graph-based metrics belong to this cluster indicating the higher importance of graph-based metrics. The Maximum complexity value-based metrics (F46, F47, F48), cohesion and coupling based metrics (F10, F54), class access type based metrics (F24, F25, F26), basic dependency-based metrics (F84, F85, F86), hub-value related metrics (71, 72) and core-value related metrics(75, 76) are some of the broad categories which predominantly constitute cluster 1.

Cluster 2 has medium priority and it consists of 23 features, 33.33% of the OO metrics and 11.54% of the graph-based metrics belong to this cluster. It is hard to identify proper groups of metrics belonging to this cluster as it lies between the cluster with the highest importance and lowest importance. However, Averages of different complexities in OO metrics (F1, F2, F3, F4), some metrics related to Lines of Code (F5, F30, F33, F56), metrics related to statements (F37, F38, F39, F40), and some of the page-rank based metrics (F67, F68) are identifiable metrics groups in this cluster.

Cluster 3 has the lowest priority and consists of 33 features 38.33% of the OO metrics and 42.31% of the graph-based metrics belong to this cluster. The metrics belonging to this cluster are seldom picked by FS techniques for training classifiers. Sum of different complexities in OO metrics (F1, F2, F3, F4, F7), instance variable-based metrics (F15, F16, F17, F19), some of the metrics based on a weighted graph representation of the software (F61, F62, F63, F64), and some of the top class-related metrics (F80, F81, F82) are a part of this cluster.

Conclusion: From Table V we can identify the features and feature groups that have more importance in SCP. 38 of the 60 OO metrics and 15 of the 26 graph-based metrics belong to the first two clusters and are frequently shortlisted by FS techniques. However, the other metrics making up cluster 3 are not necessarily irrelevant; it might also mean that the information they are conveying is also provided by other metrics.

VI. THREATS TO VALIDITY

A. Internal Validity - Internal validity deals with the bias in the various choices we make in our study that can affect the validity of the cause-and-effect relationships drawn from it. In this study, we used OO metrics and graph-based metrics as independent variables for predicting change-proneness of classes. These metrics are very popular and effective in past SCP studies. The 11 FR techniques compared in this study are taken from the various categories to make the study more generalized and extensive. The choice of NB, DTree and SVM was based on their popularity, effectiveness and diversity of prediction algorithms utilized by them. However, this can be a source of bias and hence more classifiers from categories like neural-networks, ensembles should be used.

B. Construct Validity - Threats to construct validity are concerned with bias in the experimental design used in the study. To eliminate this bias, we have described the complete experimental setup in Section IV. We used SMOTE which is a widely recommended technique to deal with class imbalance. We have mentioned the implementation details of every FR technique used in this comparative study. The parameter settings used for each of these FR techniques are carefully chosen through extensive experiments and suggestions from previous studies. However, a more careful experiment specifically for choosing those hyper-parameters where no guidelines are mentioned. We used AUC-ROC as a performance measure due to its wide usage in similar studies especially for imbalanced datasets. Nonetheless, other performance measures like F-Measure can also be employed here.

C. Conclusion Validity - Conclusion Validity deals with the reliability of the experimental observations and results drawn from the study. We have used the popular Freidman test followed by the double Scott-Knott test at a significance level of 0.05 to evaluate the performance of the FR techniques. The use of 6 diverse datasets and 3 classification techniques makes this study extensive and ensures that all the conclusions drawn in the study are significant and not just trivial effects.

D. External Validity - External Validity deals with the generalization of our experimental findings. In this study, we generated 6 new datasets derived from open-source Java projects. We collected this dataset from two sources, one is the popular proprietary tool UNDERSTAND and other is an open-source research dataset. Although the datasets are diverse in terms of the domain, code structure, size etc., we cannot claim that our findings work with datasets with a different set of metrics or if they are project-level or module-level.

VII. Conclusion And Future Work

In this study, we compared the performance of 11 FR techniques in SCP using OO and graph-based metrics. Extensive experiments were conducted on 6 newly generated Java datasets and across 3 classifiers. Using AUC-ROC performance measure along with the statistical tests for evaluation and comparison, the major conclusions drawn from the experimental findings are –

1) FR techniques are effective in SCP. They help in the extraction of features that are useful in predicting change-prone classes. The use of such features increases the predictive performance of classifiers.

2) There are significant differences in the performance of the 11 FR techniques compared in the study. While LDA, AE, and Wrapper-based FR techniques achieve better performance across different datasets and classifiers., FR techniques like PSO, PCA, and some filter-based techniques have shown unsatisfactory results.

3) The type of classifier used in SCP also affects the choice of FR techniques. The performance rankings of FR techniques like ABC and FF vary considerably with a variation in the classifier they are used with.

4) Some features have a higher probability of being chosen by FR techniques over others for model training in SCP. In the OO metric suite, coupling, cohesion, cyclomatic complexity, and lines of code are highly preferred. In graph-based metrics, inter-class dependencies and centrality measures like K-Coreness are highly effective.

In the future, we will extend this study by including more categories of FR techniques like hybrid and clustering-based techniques. The inclusion of more metrics like change-history of class and developer-related factors is another possible extension.

VIII. References

[1] K.K. Aggarwal and Y. Singh, “Software engineering”, New Age International (P) Limited, 2008.

[2] R. Malhotra and M. Khanna, “Software Change Prediction: A Systematic Review and Future Guidelines", In e-Informatica Software Engineering Journal, vol. 13, no. 1, pp. 227–259, 2019. doi: 10.5277/e-Inf190107.

[3] R. Malhotra and M. Khanna, “Mining the impact of object oriented metrics for change prediction using Machine Learning and Search-based techniques”, International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015. doi:10.1109/icacci.2015.7275614

[4] R. Malhotra and M. Khanna, “An exploratory study for software change prediction in object-oriented systems using hybridized techniques”, Automated Software Engineering, vol. 24, issue 3, pp. 673–717, 2016. doi:10.1007/s10515-016-0203-0

[5] R. Malhotra, R. and A.J. Bansal, “Software change prediction: a literature review”, International Journal of Computer Applications in Technology, vol. 54, no. 4, pp. 240-256, 2016. doi:10.1504/ijcat.2016.080487

[6] P. Bhattacharya, M. Iliofotou, I. Neamtiu and M. Faloutsos, "Graph-based analysis and prediction for software evolution," 2012 34th International Conference on Software Engineering (ICSE), pp. 419-429, 2012. doi: 10.1109/ICSE.2012.6227173.

[7] H. Kagdi, and J.I. Maletic, “ Combining Single-Version and Evolutionary Dependencies for Software-Change Prediction”, Fourth International Workshop on Mining Software Repositories, 2007. doi:10.1109/msr.2007.2

[8] G. Catolino, F. Palomba, A. De Lucia, F. Ferrucci, and A. Zaidman, “Enhancing change prediction models using developer-related factors”, Journal of Systems and Software, 143, pp. 14–28, 2018. doi:10.1016/j.jss.2018.05.003

[9] M. Jureczko, “Significance of Different Software Metrics in Defect Prediction”, Software engineering : an international Journal (SeiJ), Vol. 1, no. 1, 2011.

[10] S. Chidamber and C. Kemerer, “A metric suite for object oriented design,” IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, 1994.

[11] Z. Xu, J. Liu, Z. Yang, G. An and X. Jia, “The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison”, IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 2016. doi:10.1109/issre.2016.13

[12] L. Kumar, S. Lal, A. Goyal, and N. Murthy, “Change-proneness of object-oriented software using combination of feature selection techniques and ensemble learning techniques,” in Proceedings of the 12th Innovations on Software Engineering Conference. ACM, pp. 1-11, 2019. doi: 10.1145/3299771.3299778

[13] 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, pp. 273–286, 2013. doi: 10.1007/s13042-012-0095-7

[14] G. Catolino and F. Ferrucci, “An extensive evaluation of ensemble techniques for software change prediction,” Journal of Software: Evolution and Process, vol. 31, issue 9, 2019. doi: 10.1002/smr.2156

[15] R. Malhotra and M. Khanna, “Software change prediction using voting particle swarm optimization based ensemble classifier”, in Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, pp. 311-312, 2017. doi: 10.1145/3067695.3076007

[16] L. Kumar, R.K. Behera, S. Rath, and A. Sureka, “Transfer learning for cross-project change-proneness prediction in object-oriented software systems: A feasibility analysis,” ACM SIGSOFT Software Engineering Notes, vol. 42, no. 3, pp. 1–11, 2017. doi: 10.1145/3127360.3127368

[17] R. Malhotra and A. Bansal, "Predicting change using software metrics: A review", 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), pp. 1-6, 2015. doi: 10.1109/ICRITO.2015.7359253.

[18] M. Lorenz and J. Kidd, “Object-oriented software metrics: A practical guide,” Prentice-Hall, Inc., 1994.

[19] F. B. Abreu and R. Carapuça, "Object-oriented software engineering: Measuring and controlling the development process," in proceedings of the 4th International Conference on Software Quality, 1994.

[20] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality as-sessment," IEEE Transactions on Software Engineering, vol. 28, pp. 4-17, 2002.

[21] Y. Yang, J. Ai and F. Wang, "Defect Prediction Based on the Characteristics of Multilayer Structure of Software Network," IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 27-34, 2018. doi: 10.1109/QRS-C.2018.00019

[22] E. Giger, M. Pinzger and H.C. Gall, (2012), “Can we predict types of code changes? An empirical analysis”, 9th IEEE Working Conference on Mining Software Repositories (MSR), 2012. doi:10.1109/msr.2012.6224284

[23] T. Quah and M. Thwin, “Application of neural networks for software quality prediction using object-oriented metrics”, International Conference on Software Maintenance (ICSM), 2003. doi:10.1109/icsm.2003.1235412

[24] S. Shivaji, S, E.J. Whitehead, R. Akella and S. Kim, “Reducing Features to Improve Code Change-Based Bug Prediction”, IEEE Transactions on Software Engineering, vol. 39, no. 4, pp. 552–569, 2012.

[25] Z. Xu, J. Liu, Z. Yang, G. An and X. Jia, “The Impact of Feature Selection on Defect Prediction Performance: An Empirical

Comparison", IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 2016. doi:10.1109/issre.2016.13

[26] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning", Artificial Intelligence, vol. 97, issues 1–2, pp. 245-271, 1997. doi: 10.1016/S0004-3702(97)00063-5

[27] J. Tang, S. Alelyani and H. Liu, "Feature Selection for Classification: A Review" in Data Classification: Algorithms and Applications, CRC Press, 2014.

[28] G. Chandrashekar and F. Sahin, "A survey on feature selection methods", Computers & Electrical Engineering, vol. 40, issue 1,pp. 16-28, 2014. doi : 10.1016/j.compeleceng.2013.11.024.

[29] W. Duch, "Filter Methods", In: Guyon I., Nikravesh M., Gunn S., Zadeh L.A. (eds) Feature Extraction. Studies in Fuzziness and Soft Computing, vol. 207, 2006. doi: 10.1007/978-3-540-35488-8_4

[30] R. Malhotra and K. Khan, "A Study on Software Defect Prediction using Feature Extraction Techniques", 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020. doi:10.1109/icrito48877.2020.9197999

[31] M. Schiezaro and H. Pedrini, "Data feature selection based on Artificial Bee Colony algorithm", J Image Video Proc, vol. 47, 2013. doi: 10.1186/1687-5281-2013-47

[32] M. Anbu and G.S. Anandha Mala, "Feature selection using firefly algorithm in software defect prediction", Cluster Computing, pp. 10925–10934, 2017. doi:10.1007/s10586-017-1235-3

[33] I. Sora and C. Chirila, "Finding key classes in object-oriented software systems by techniques based on static analysis", Information and Software Technology, vol. 116, 2019. doi: 10.1016/j.infsof.2019.106176

[34] [dataset] https://figshare.com/s/df5529168b9641bdd96e

[35] A. Balogun, S. Basri, S. J. Abdulkadir, V.Adeyemo, "Software Defect Prediction: Analysis of Class Imbalance and Performance Stability", Journal of Engineering Science and Technology 14(6), pp 3294-3308, 2019.

[36] E. Alpaydin, Introduction to Machine Learning, Cambridge, MA, USA: MIT press, 2014.

[37] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings", The Annals of Mathematical Statistics, vol. 11, pp. 86–92, 1940. doi: 10.1214/aoms/117773194

[38] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models", In Proceedings of the 37th International Conference on Software Engineering (ICSE), pp. 789-800, 2015. doi: 10.1109/ICSE.2015.91

[39] N. V. Chawla , K. W. Bowyer , L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", Journal of Artificial Intelligence Research, vol.16, pp. 321-357, 2002. doi: 10.1613/jair.953

[40] M. Lv, Y. Ren, and Y.Chen, "Research on imbalanced data based on SMOTE-AdaBoost algorithm", 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), 2019. doi: 10.1109/EITCE47263.2019.9094859

[41] M. Cherrington, F. Thabtah, J. Lu, and Q. Xu, "Feature Selection: Filter Methods Performance Challenges", 2019 International Conference on Computer and Information Sciences (ICCIS), 2019. doi:10.1109/iccisci.2019.8716478

[42] S. Valle, W. Li, and S.J. Qin, "Selection of the Number of Principal Components: The Variance of the Reconstruction Error Criterion with a Comparison to Other Methods", Industrial & Engineering Chemistry Research, 38(11), pp 4389–4401, 1999. doi:10.1021/ie990110i